Oliver Pajonk,

Overview of System Identification with Focus on Inverse Modeling : Literature Review

Braunschweig : Institut für Wissenschaftliches Rechnen, 2009

Informatikbericht / Technische Universität Braunschweig ; Nr. 2009-03

Veröffentlicht: 14.07.2009

http://www.digibib.tu-bs.de/?docid=00028636

Oliver Pajonk

Overview of System Identification with Focus on Inverse Modeling

Literature Review

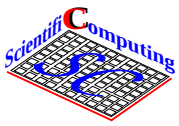
Institute of Scientific Computing
Carl-Friedrich-Gauss-Fakultät
Technische Universität Braunschweig

Braunschweig, Germany, 2009

Pajonk, Oliver: Overview of System Identification with Focus on Inverse Modeling
Informatikbericht 2009-03, Technische Universität Braunschweig
Version: 1.0 (Build Version 232)

This document was created on July 14, 2009 using LaTeX 2_ε.

Institute of Scientific Computing
Technical University Braunschweig
Hans-Sommer-Strasse 65
D–38106 Braunschweig, Germany



Contents

List of Figures

Foreword

The intention behind this literature review is to obtain knowledge about the current status in the field of system identification with special focus put on the inverse modelling step. There the parameters for a model are to be determined by taking data obtained from the true system into account.

The application in mind is located in geophysics, especially oil reservoir engineering, so special focus is put on methods which are relevant for system identification problems that arise in that context. Nonetheless the review should be interesting for everybody who works on system identification problems.

Chapter 1

Introduction

This literature review gives an overview of the current state of system identification methods. The focus is put on the step of *inverse modelling* which aims to identify system parameter values from real measurements of the system of interest.

This introduction can be seen as an extended glossary that presents the vocabulary used in the following chapters and discusses its sometimes inconsistent usage in the literature.

1.1 What is a System?

Most authors in system identification and other scientific fields do not employ a formal definition of the concept of a *system*. Many use natural language definitions or descriptions like the following:

> A system may be defined as a set of elements standing in interrelation among themselves and with environment. [vB68]

> A system is a set of entities with relations between them. [Lan95, p. 55]

> Nach DIN 66201 wird unter einem *System* eine abgegrenzte Anordnung von aufeinander einwirkenden Gebilden verstanden. [Ise92, p. 1]

The following is essentially a translation of the previous definition – it is from the same author, but between these two references are 13 years of research.

> A *system* is a bounded arrangement of formations influencing one another. [Ise05, p. 33]

> In loose terms a *system* is an object in which variables of different kinds interact and produce observable results. [Lju06, p. 1]

Some, like [Tar04], simply use the word "system" without any explanation.

This implicit usage may be due to the fact that we have an intuitive understanding of the word "system" as it is ubiquitous in everyday language: solar system, computer system, social system, political system, nervous system, traffic guidance system and many others. This is supported by [Lju06] who states:

> Instead of attempting a formal definition of the system concept, we shall illustrate it by a few examples. [Lju06, p. 1]

Another reason may be the fact that even scientific areas like *cybernetics*[1] and *general systems theory* have not produced a commonly accepted formal definition of the word "system". [Sky07, p. 56-75] discusses the meaning of "system" on 19 pages!

The need to formally define the concept of a "system" has also been recognised by other authors, for example [Mar75] and [Bac00]. The latter one I present as an example for a formal definition in the following section, whereas the prior one contains an illuminating discussion of the criticism that a generic description of "system" is vacuous (a opinion that the author of that article does not share).

1.1.1 Formal Definitions and Some Remarks

[Bac00] discusses some earlier definitions of the concept of a system and explains their formal weaknesses. He also introduces interesting abstract definitions of system, super- and subsystem, open and closed system which I will present now with some simplifications[2]:

Definition 1.1. A *system* $S = (E, R)$ is a tuple consisting of a set of *entities* called E and a non-empty set of *relations* on these entities R, satisfying the following conditions:

1. $|E| \geq 2$.
2. The *degree* of all relations $R_i \in R$ is $\deg(R_i) \geq 2$ (the degree of a relation states how many entities are related by this relation).
3. There is a *path* from a to b if $(a, b) \in R_1$, where $R_1 \in R$ (here: $\deg(R_1) = 2$).
4. There is a *path* from a to b *and vice versa* if $a \in \{a_1, a_2, \ldots, a_n\} \wedge b \in \{a_1, a_2, \ldots, a_n\} \wedge (a_1, a_2, \ldots, a_n) \in R_2$ for some $R_2 \in R$ and $a_1, a_2, \ldots, a_n \in E$ (here: $\deg(R_2) \geq 3$).
5. There is a *path* from a to b if there is an entity $c \in E$ such that there is a path from a to c and there is a path from c to b (transitive hull).
6. From every entity in E there is a path to every other entity in E.

Remark 1.1. Note that this definition makes a difference between binary relations ($n = 2$), which are *directed*, and relations of higher degree ($n \geq 3$), which are *undirected*: if $(a, b) \in R_1$ there is a path from a to b but not necessarily from b to a. But if $(a, b, c) \in R_2$ there is a path from each entity to every other entity, for example from a to b and from b to a. This approach is debatable, of course: if we have a relation between a and b - doesn't this imply a relation from b to a in any case?

Definition 1.2. Let $S_1 = (E_1, R_1)$ and $S_2 = (E_2, R_2)$ be two systems. Then S_1 is a *subsystem* of S_2 iff

- $S_1 \in E_2$

or

- $S_1 \neq S_2$ and
- $E_1 \subseteq E_2$ and
- for every relation $r \in R_1$ there is a relation $s \in R_2$ such that $r \subseteq s$ is true.

A system S_1 is a *suprasystem* / *supersystem* of a system S_2 iff S_2 is a subsystem of S_1.

[1] "Cybernetics is the interdisciplinary study of the structure of complex systems, especially communication processes, control mechanisms and feedback principles.", [Wik08].

[2] He does not use the word "entity" for the members of the set of interest – maybe to avoid premature specialisation – but the word is inherently unspecialised (see, for example, http://en.wikipedia.org/wiki/Entity), so we use it in our definition. This is in accordance with [Lan95].

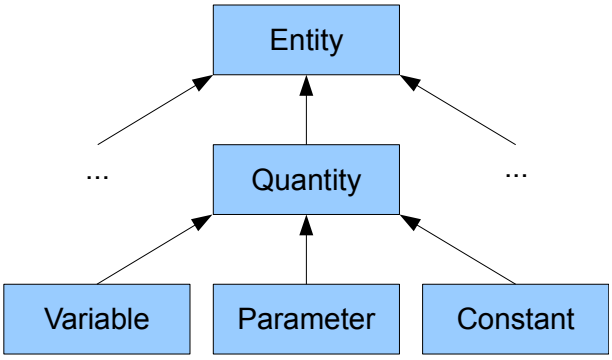


Figure 1.1: The Different Concepts of Quantites

Definition 1.3. A system S_1 *interacts with / affects / has a relation to* another system S_2 iff there is a relation between at least one element in S_1 and at least one element in S_2. A *closed* system is a system which has no relation to any other system that is not a subsystem of it and to which no other system that is not a subsystem of it has any relation. An *open* system is a system which is not closed.

Remark 1.2. The concept of a *system boundary* is not introduced by him explicitly but he makes implicit use of it in his example graphs. Its intuitive meaning is clear: a system S is surrounded by its boundary. Relations to other systems can cross this boundary - then it is an open system. See also [Sky07, p. 62] for a discussion on this topic.

Above example should just give you an impression how such of a formal definition of "system" could look like. Instead of using on this definition for the rest of this review I will also rely on the intuition of the reader and continue with the introduction of additional concepts that I will use throughout the text. The only two important concepts that I will carry over from this section are *entities* and *relations*, as in the definition of [Lan95, p. 55] (see page 3).

1.1.2 Additional Concepts

Supplementing the understanding of a system I also need to mention and explain some additional, important concepts that appear throughout the literature.

1.1.2.1 Quantities, Variables, Parameters and Constants

Quantities are special entities that have a *magnitude* (a continuous value describing a continuum) or *multitude* (a discontinuous, countable value). In systems, quantities appear in three different concepts (see also figure 1.1):

Variables are quantities that can change their value. An example is the x in $f(x) = x^2$.

Parameters are quantities that define certain characteristics of a system. They can be changed, but changing them means obtaining a different system. An example is the a in $f(x) = x^a$: for each a we obtain a very different behaviour of the system. But we can also see here that there is not necessarily a clean distinction between parameters and variables: we could easily write it the other way around $f(x) = a^x$. But then again this describes a very different system.

Constants are quantities that can never change their value. An example is π.

Often quantities have an associated meaning or interpretation within the system. They describe a *measure* of something within the system. This meaning is often clarified by a concept called *unit of measurement*, which is associated with the quantity.

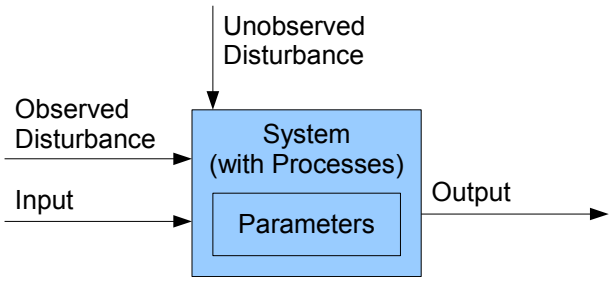


Figure 1.2: A System with Input, Output, Disturbances and Process

Remark 1.3. Some authors are sloppy in their usage of terms or simply have a different meaning associated with their terms. [Tar04] for example uses "parameter" where according to my understanding the word "variable" would be appropriate, as in "observable parameter". This does not invalidate his reasoning, it only makes discussing it more difficult.

Remark 1.4. Parameters and constants are sometimes collected into the class of *exogenous* variables.

1.1.2.2 Input, Output, Disturbance and Processes

A system can be seen as a black box which somehow transforms some quantities into other quantities. Especially in signal transmission this is the usual model for signal processing systems. The quantities in this case represent signals. This transformation is usually called a *process*. In the following I will explain these connected concepts in more details. See also figure 1.2 for a graphical overview.

Input Some non-constant quantities of a system may be independent from any other part of the system, so that we can freely choose their values (maybe within some physical boundaries). These quantities in turn influence other entities in the system through their relations to them and by this influence the system as a whole. Then they are regarded as the system *input*. If they are variables they are also called *independent variables* or simply *input variables*.

Output If some variables of a system depend on and are determined by the input and the system itself and we can observe their values we call them system *output*. For obvious reasons these variables are also called *dependent variables*.

Remark 1.5. Constants can be seen as part of the input, but never as part of the output - they cannot depend on other entities, as they are, well, *constant*. Parameters may also be seen as part of the input, but as they describe system characteristics they are usually seen as part of the system itself. To my opinion this is the appropriate way because they normally describe an integral part of the system, whereas the input "crosses the system boundary" and describes external influence. The concept of parameters will become clearer when we talk about forward and inverse modelling, where parameters are of primary concern.

Disturbance Input variables whose values cannot be chosen freely are usually called *disturbance* [Lju06, p. 1]. We can divide them into those disturbances which can be measured directly and those which can only be observed by their influence on the system output. It is important to note that disturbance can be often quantified by its statistical properties.

An example for observable disturbance is the solar radiation going into a solar heated house [Lju06]. We cannot influence the amount of radiation directly in a reasonable way, but we can measure it before it enters the system.

An example for an unobservable disturbance is acoustic noise coming from the speakers of a sound system. The source of the noise may be thermal noise in the components of the amplifier,

which cannot be observed (measured) directly. But we can certainly observe their influence on the output.

Remark 1.6. The distinction between input and observable disturbance is often unclear, but also not of great importance [Lju06, p. 1]. It is only a concept that should help to distinguish different types of system input and simplify discussions about it.

Process The transformation of input quantities to output variables is usually called *process*. For example the transmission of a audio signal over a copper cable can be modelled in such a way. The cumulative processes in a system can be seen as the reaction of the system to the environment (its input), which is called the *behavior* of a system.

Remark 1.7. To show another example for the inconsistent usage of the words explained in this chapter I will give a quote from [Nel01, p. 18]:

> *Process:* Used as a synonym for the plant or system under study. It can be static or dynamic.

So attention must be paid on the meaning of the words used in literature that has the word "system" in its title.

1.1.2.3 State and Dynamic Systems

In most literature on system identification the considered systems are so-called *dynamic* systems. The concept of dynamics is inherently coupled with a concept called *state*. A system which has a state is a system that contains variables whose values depend not only on the current input of the system but also on previous input [Lju06, p. 5]. This implies that we are introducing the dimension *time* in our system.

These type of variables are called *state variables*. They are not part of the input as they depend on other entities in the system, although they are often treated that way. But if they are observable they can be part of the system output, either directly or hidden behind a measurement function and possibly obscured by measurement noise. Note that some authors call them *state parameters* but I will stick with the name state variable here as it seems more appropriate.

Dynamic systems are also called *time variant* systems - their behaviour changes with time: they give different output for the same input at different times.

Outputs of dynamic systems whose inputs are not observed are often called *time series* [Lju06, p. 5].

1.1.2.4 Phase Space

The notion of "state" motivates the concept of a *phase space*. This is a space where all possible states of a dynamic system are represented by unique points. Thus a certain trajectory in this phase space can represent the time evolution of a system state on which a certain process acts.

1.1.2.5 Purpose

There are some discussions if having a *purpose* or being *goal-oriented* is part of the definition of a system. Such philosophical questions are out of scope for this review and are thus not considered. Just a single remark is made to set you thinking about this question: which purpose has the solar system?

1.1.3 Examples of Systems

To fill the previous definitions with meaning we present some example systems. More examples can be found, for example, in [Lju06, p. 2–5].

1.1.3.1 Automobile

The automobile is a complex technological system that we use in workaday life, so it will make an excellent example for some of above definitions.

- Some of the *entities* in the car are $E = \{$ motor, car body, tires, wheels, $\ldots \}$.
- Some of the *relations* are $R = \{$ is attached to, transmits force to, $\ldots \}$.
- The entity "motor" can certainly be seen as a *subsystem* of the automobile; possible entities are $E = \{$ cylinder block, pistons, crankshaft, valves, $\ldots \}$. A relation going across the *boundary* of the system "motor" to its *supersystem* "automobile" is certainly the "is attached to" relation.

This example shows us an important feature of a "system": most of them are hierarchical and it makes sense to consider subsystems as well as supersystems.

1.1.3.2 Democratic Governance System

Abstract systems like political or social ones are not so easy to grasp, so we will give one example.

- Some of the *entities* in the democratic governance system are $E = \{$ executive, legislature, judiciary, auditory, constitution, constitutional court, president, people, weapons, $\ldots \}$.
- Some of the *relations* are $R = \{$ is based on, creates, controls, is part of, uses, $\ldots \}$.

In this example we see that within the same system very different entities (persons, abstract objects, real things, ...) and very different relations between them can exists.

1.1.3.3 Solar System

The next example is a physical or natural system, as opposed to the other two examples.

- Some of the *entities* in the solar system are $E = \{$ our planets, sun, (other planets), (galaxies), asteroids, comets, dust, moons, $\ldots \}$.
- Some of the *relations* are $R = \{$ gravity, radiation, orbit, $\ldots \}$.

This example should point you to a very important question: when we consider the "solar system" - do we have to put in entities like "other planets" or "galaxies"? Do we have to put in "asteroids" - which in fact do exist within the (physical) boundaries of the solar system? Do we have to consider "radiation" - or can we *leave them out because their influence is negligible for our purpose*? See section 3.1.1 for further discussion on this topic.

1.2 What is System Identification?

The discipline of system identification tries to build *mathematical models* of systems for a certain purpose, guided by measurements. These models are different from others like *mental models* or *graphical models* (see [Lju06, p. 6]): as we can evaluate them using a computer they tend to be highly complex.

Thus creating such a model can be a very challenging task: we have seen that many different types of systems exist and that there is essentially no limit in complexity for both the relations inside the system and the entities that interact in the system. So the mathematical models that we create to describe certain aspects of a system can become arbitrarily complex, too.

The problem of system identification is also pervasive in science and engineering; thus many different applications have resulted in a multitude of different approaches, model types and methods to solve the problem. Some are specific for the respective application, some have a broader use.

1.2.1 Brief History

As it is the case with many modern methods, also the roots of system identification can be traced back quite a long time. For example the famous Gauss-Newton-method was developed by Carl Friedrich Gauss to solve the system of equations arising in his non-linear least squares method for regression. With the help of this method he found values of parameters in a model of the trajectory for the dwarf planet Ceres - which clearly is a system identification problem. This happend between 1795 and 1802, so more than 200 years ago. The method of least squares Gauss developed is well known and, of course, still in use.

It took another 150 years before the advent of electronic computers, and with it came the rise of (computational) system identification. According to [Gev06] the modern discipline of system identification started around 1960 as part of control theory. It was part of model-based control design, which was very much en vogue at that time due to the development of the Kalman filter.

System identification went from deterministic methods to maximum likelihood methods and finally to stochastic methods until the 1970ies. A confusing amount of approaches was developed until that time, so that structuring them became more and more necessary. The inevitable cleanup period of 1975 to 1985 resulted in the first edition of [Lju06], which was released in 1987 - together with a MATLAB toolbox for system identification.

Then important steps were made in closed-loop identification, subspace based identification and non-linear identification. According to [Gev06], which contains a lot more detailed historical account of system identification, this is the current state of development.

1.2.2 Other Types of Models

As mentioned before, system identification deals with mathematical models. These I understand as abstract descriptions of the target system in the mathematical notation and language. If no closed, analytical solution for the model can be found with mathematical methods (which is often the case, especially for real world problems) such descriptions can then be transformed with the help of computer science into efficient computer programs to perform real computations with the model.

But there are also other types of models, which include but are not limited to the following:

Mental models To be able to, for example, walk, a person has to have a mental model of its body[3]. The person can then use this model to identify which muscles to use when, for example, a left turn has to be performed. Another example of a mental model is the model of a car created by driving lessons. Afterwards the driver knows what to do when he wants to influence the car such that it, for example, turns left or drives approximately 30 km/h.

Graphical models Some systems can be described in an adequate way by a graphical model, like a drawing or a table. An example is a linear system, which is uniquely described by its impulse or step response [Lju06, p. 6].

Physical models To explain a desease to his patient, a doctor may use a plastic model of a body part. Physical models are also used in wind tunnels to design and verify the shape of cars or aircrafts.

We see that these types of models are an integral part of our day-to-day life. Mathematical models on the other hand tend to be more complex, more exact but less accessible to the majority of people.

1.2.3 Uses of System Identification

There are several use cases for system identification procedures in industry and science. Some are named by [Nel01]:

[3] How this model is created is not important here; but a large amount of it is certainly created by learning.

System Analysis We want to obtain more insight in a certain system.

Prediction Given the current state of a system we want to be able to predict the behaviour of a certain system. The popular example is weather forecasting.

Simulation We want to simulate the behaviour of a system given only the input.

Optimisation We want to optimise a certain aspect of the true system but we cannot operate directly on it to find this optimum. Reasons may be cost, safety, security or time constraints.

Control We want to develop an advanced controller for a real process which involves a model of that process.

Fault Detection We want to be able to detect false behaviour of a true system by comparing the model output with the output of the true system.

Chapter 2

Formulation of the Problem

The problem that we want to solve during the system identification process can be described as follows: for a certain "real system"[1], which we call $\tilde{G}$ in this review, we need, for a certain reason[2], a mathematical model. We additionally have a set of measurements for this "real system". Such measurements can consist of measurements of output variables and parameters, together with associated measurements of the input variables and state variables. Sometimes the measurements are incomplete, sometimes biased, and often noisy. This total set of measurements we will conveniently call *data* from now on.

The problem we have to solve now is the following: how can we create a mathematical model for the "real system" $\tilde{G}$, *guided by available data*, such that it is "good enough" for our purpose?

To introduce a mathematical notation we will now formulate the problem in a more formal way: we are in search of a *mathematical model operator* G that approximates the real model operator $\tilde{G}$ in such a way that

$$\mathbf{d} = G(\mathbf{x}, \mathbf{m}) \tag{2.1}$$

and $\|\mathbf{d} - \tilde{\mathbf{d}}\|$ is small in some norm for all possible input/output combinations[3]. The letters represent throughout the review the following quantities:

- $\mathbf{x}$ are input variables, including observable disturbances,
- $\mathbf{m}$ are model parameters, including non-observable disturbances,
- $\mathbf{d}$ are outputs of the model and
- $\tilde{\mathbf{d}}$ are the measured outputs of the real system.

In addition to the model operator definition and parameter estimation problems the considered system may be time dependent. Remembering from section 1.1.2.3 that this introduces a state into the model we see that the additional problem of correctly estimating the model state variables may also be put on us.

This system identification problem came up in many quite different contexts during history. Thus several views on the problem have developed, which I will present to you now.

[1] By this I mean a certain part of reality that has not yet undergone any modelling. So strictly speaking it is not yet a system, as part of system identification is the extraction of the system from its surroundings.

[2] Some possible reasons are given in section 1.2.3.

[3] Here we will not define from which spaces the operator and the quantities are as this should only motivate a mathematical view.

2.1 Three Different Views on the Problem

I have sorted the different views into three main categories. The most generic view we can take on the system identification problem is a fully probabilistic one, which is the first one I will present now. The other two views are motivated from a more practical point of view and came up in different application fields.

All views are somehow mutually connected, of course, as they essentially deal with the same problem. This becomes especially apparent when some special assumptions on the involved uncertainties are made. Connections will be pointed out where appropriate.

2.1.1 Probabilistic View

The most generic view we can take is to regard the set of conceivable models of the system and the set of possible measurements as abstract *manifolds*, which is done in [Tar04, p. 1–40][4] or [MT02]. They introduce probability densities on these manifolds, which they then use to formulate the "state of information" the researcher has with regard to the problem: the physical model itself, which connects the input[5], the parameters and the output[6], is a probability distribution which accounts for uncertainties like the lack of knowledge of the researcher or imperfect parameterisation. This probability density they call the "theoretical probability density".

The solution of the inverse modelling problem is introduced as a probability density, too: the conjunction of prior knowledge ρ with the theoretical probability density θ gives the posterior probability density σ through the updating equation 2.2:

$$\sigma(\mathbf{d},\mathbf{x},\mathbf{m}) = k\frac{\rho(\mathbf{d},\mathbf{x},\mathbf{m})\theta(\mathbf{d},\mathbf{x},\mathbf{m})}{\mu(\mathbf{d},\mathbf{x},\mathbf{m})}. \tag{2.2}$$

Here $\mu(\mathbf{d},\mathbf{x},\mathbf{m})$ represents the so-called homogeneous state of information and k is a normalising constant. This equation is a slightly modified version of equation (1.83) of [Tar04, p. 32], adopted to my notation. It solves a very general problem, which is specialised for the more common particular cases in [Tar04, p. 33ff]. The usage of the model operator G is "hidden" in the pdf θ, where it relates possible inputs and models to possible simulated measurements.

The[7] solution to inverse problems (the specialisation of equation 2.2 which is of importance in this review) is given by

$$\begin{aligned}\sigma_M(\mathbf{m},\mathbf{x}) &= k\rho_M(\mathbf{m},\mathbf{x})\int_{\mathfrak{D}}\frac{\rho_D(\mathbf{d})\,\theta(\mathbf{d}|\mathbf{m},\mathbf{x})}{\mu_D(\mathbf{d})}\,d\mathbf{d}\\ &= k\rho_M(\mathbf{m},\mathbf{x})\,L(\mathbf{m},\mathbf{x}).\end{aligned} \tag{2.3}$$

Here $\sigma_M(\mathbf{m},\mathbf{x})$ is the posterior information on the model- and input variable space (the updated state of information), $\rho_M(\mathbf{m},\mathbf{x})$ is the prior information on the model- and input variable space, $\rho_D(\mathbf{d})$ is the prior information on the output variable space (data space, measurement space) and $\theta(\mathbf{d}|\mathbf{m},\mathbf{x})$ is the theoretical information on the output variable space, given the model- and input variables. $\mu_D(\mathbf{d})$ is the homogeneous state of information on the output variable space and k is a constant. The integral part is also called the *model likelihood function* for $\mathbf{m},\mathbf{x}$, which we abbreviate as $L(\mathbf{m},\mathbf{x})$. We can describe the equation in a sentence: the updated state of information is the prior state of information times the model- and input data likelihood (given the measured output data of the real system), times a constant.

In practical problems the modelisation uncertainties can often be neglected [Tar04, p. 34f]. Then we have[8] $\mu_D(\mathbf{d}) = const$ and $\theta(\mathbf{d}|\mathbf{m},\mathbf{x}) = \delta(\mathbf{d}-G(\mathbf{m},\mathbf{x}))$, where G is again the mathematical

[4] Note that this nice book is generously made available for free by the author as PDF download from his webpage. See http://www.ipgp.jussieu.fr/~tarantola.

[5] They do not distinguish between input and parameters and call them "model parameters".

[6] In their notation the measured system output is simply called "measurements".

[7] It is really *the* solution. Please see [Tar04, p. 33f] for the reasoning and why the pdfs can be modified like this.

[8] Here we have to assume that the data space is a linear space. See [Tar04, p. 34] for details.

model operator from equation 2.1 on page 11. Then the likelihood function simplifies to $L(\mathbf{m}, \mathbf{x}) = \rho_D(G(\mathbf{m}, \mathbf{x}))$, so all in all our solution equation 2.3 on the preceding page becomes in this case

$$\sigma_M(\mathbf{m}, \mathbf{x}) = k\rho_M(\mathbf{m}, \mathbf{x})\, \rho_D(G(\mathbf{m}, \mathbf{x})). \tag{2.4}$$

That equation avoids the computation of an integral over the data space to evaluate the model likelihood function.

To summarize we can say that the central focus of the probabilistic view is to compute $\sigma_M(\mathbf{x}, \mathbf{m})$ (generally: $\sigma(\mathbf{d}, \mathbf{x}, \mathbf{m})$, but this is not necessary for solving equation 2.3), the pdf representing the best possible solution to the inverse problem, as exact as possible (given the data).

2.1.2 Ill-posed Optimisation Problem View

The ill-posed optimisation problem view (as I call it) works in a quite different way. If we look again at equation 2.1 we can quickly identify the solution with a "simple" algebraic manipulation. Given the measured data $\tilde{\mathbf{d}}$ of the real system we can write it down as:

$$(\mathbf{x}, \mathbf{m}) = G^{-1}(\tilde{\mathbf{d}}) \tag{2.5}$$

We simply use the inverse model operator to compute the model and the input from measurements. Unfortunately this solution has two problems.

The first is that the (hypothetical) inverse model operator G^{-1} does not generally depend continuously on the data. Changing the input $\tilde{\mathbf{d}}$ slightly can result in very different output $\mathbf{x}, \mathbf{m}$, meaning in very different models. It is also often the case that different choices for $\mathbf{m}$ result in the same model output $\mathbf{d}$, meaning that the operator G is not injective. This becomes clear if we think about the size of the spaces from which $\mathbf{x}, \mathbf{m}$ and $\mathbf{d}$ come from: the model space is generally much larger than the measurement space. This property is typical for so-called *inverse problems*, which are ill-posed in the sense of Hadamard [Kir96, p, 10]. [DES98] calls G a smoothing operator, so G^{-1} is a rough operator (loosely speaking) which has exactly the problem of not depending continuously on the data (see also [Kal97]).

The second problem we have to consider is that the measured data $\tilde{\mathbf{d}}$ stems from a real physical process and was obtained using a device with limited accuracy. Thus it always contains measurement errors of a certain magnitude so that we cannot fully trust it. This has to be taken into account by the method, too.

Together these two problems impose serious difficulties in solving equation 2.5 directly. Simply computing $\min_{\mathbf{x},\mathbf{m}} \|G(\mathbf{x}, \mathbf{m}) - \tilde{\mathbf{d}}\|$ in some norm using some optimisation procedure does not always lead to a satisfying result, so we have to employ special methods.

2.1.3 State Space View

The main focus of the state space view is quite different to the previous views as it is historically concerned with correctly estimating *state variables* $\mathbf{s}_t$ of a *dynamic* system model, not static parameters $\mathbf{m}$ of the model. Thus it can be seen as a complementary extension of these two views. But as it came up in many applications and independently from the previous two views I treat it separately.

An important assumption which is commonly applied in the state space view is that the underlying process is a *Markov process*[9], meaning that the state of the current time step, $\mathbf{s}_t$, only depends on the state of the previous time step, $\mathbf{s}_{t-1}$.

[9] It is possible to consider also non-Markovian processes in a probabilistic setting, but it is not very common [DdFG01, p. 5].

The state space view on the problem can then be conveniently formulated using probability densities [DdFG01, p. 5], similar to the probabilistic view of section 2.1.1:

$$p(\mathbf{s}_0) \tag{2.6}$$

$$p(\mathbf{s}_t|\mathbf{s}_{t-1}) \qquad \text{for } t \geq 1 \tag{2.7}$$

$$p(\mathbf{d}_t|\mathbf{s}_t) \qquad \text{for } t \geq 1 \tag{2.8}$$

Here $p(\mathbf{s}_0)$ is the pdf of the initial state (which is usually guessed or somehow given), $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ is the so-called transition equation and $p(\mathbf{d}_t|\mathbf{s}_t)$ desribes how measurements are related to the state. For these last two equations practical computations are performed using the model operator G. Model parameters and input variables are not mentioned in this formulation but they can additionally be considered in the same way as in the probabilistic view. The solution to compute updated pdfs of the state variables and model parameters from measurements is then also given by applying the Bayesian theorem.

What sets apart this view and the methods that result from it is the pervasiveness of the time dimension: all methods exploit it and make *sequential* updates to model states (and parameters, if any) along with the simulation runs. They are able to process information as it arrives. This quite different to the other two views where a time dimension does not appear in general – especially in the ill-posed optimisation problem view.

2.1.4 Discussion

The two most widely applied views are certainly the ill-posed optimisation problem view and the state space view. Both are quite different, though, and originate from different application fields. The more general, *fully* probabilistic view is acknowledged by other scientists but not really exploited and propagated: its great generality and complexity limits its usability for practical purposes.

2.2 Uncertainty Quantification

Is is often not sufficient to find a single model (a single set of parameters) as solution to the system identification problem. The problem is that the data that we use to identify the parameters is generally uncertain, too: it contains measurement errors. Together with the fact that the solution of the inverse problem stated in equation 2.5 does not depend continuously on this data this forces us to give, at least, a "best" solution and error bars on this solution. Better would be to compute a full probability density function on the model parameter space, of course. This would enable us to assess the quality of the solution, relate it to other uncertain data and depend possible decisions on this.

However this uncertainty quantification requires considerably more computational resources and was thus only used for small models. But together with the ever increasing computation power this type of uncertainty quantification comes more and more into reach, even for real world problems. Especially the methods employed in the state space view also incorporate aspects of uncertainty quantification.

Chapter 3

System Identification Process

The process of system identification can be divided[1] into the following steps, but, like [Tar04] puts it, there exists "strong feedback [...] between these steps, and a dramatic advance in one of them is usually followed by advances in the other two.". See also figure 3.1.

(1) Simplification Put a system boundary with the purpose of the model in mind.

(2) Quantification Find a minimal set of *model quantities* that fully describe the system. This especially includes the *model parameters* (see section 1.1.2 for a definition of model parameters), so this step is often called *parameterisation*.

(3) Forward modelling Employment of mathematical formulae that can be used to make *simulations* for the system output given values for the model parameters and the input.

(4) Inverse modelling Obtain actual values for the model parameters when given some values for the output of the *real* system, obtained by measurements.

This division essentially stems from [Tar04, p. 1-2], but he only mentions the last three steps and misses the first one. In my opinion this is a very important step where we create the first model errors by *simplification*: we leave out or simplify relations of the system that seem to be of minor relevance.

Additionally it is important to mention that these steps are iterative, of course: if we identify problems in a later step that were caused by a previous one we have to go back to fix them. This possibly means re-doing those steps. In figure 3.1 this is depicted by the arrows connecting the steps.

The focus of this review is on the methods used in step (4). We will describe the basics for each step later in this chapter, describe the general process and thus motivate an overview. The following chapters will then treat certain aspects, especially of step (4), in more details.

Of course there are other divisions of the system identification process (see also section 3.2), but in this review I will stick with above division as it seems most appropriate to me.

3.1 The Four Steps Explained

3.1.1 Step (1): Simplification

The question whether an influence (a relation!) from an entity is important, simplifyable or negligible is an important and non-trivial question. It stands at the beginning of every system identification process.

[1] This division is quite arbitrary, but it fits our needs. Other authors mention different divisions that fit their purposes. See also section 3.2.

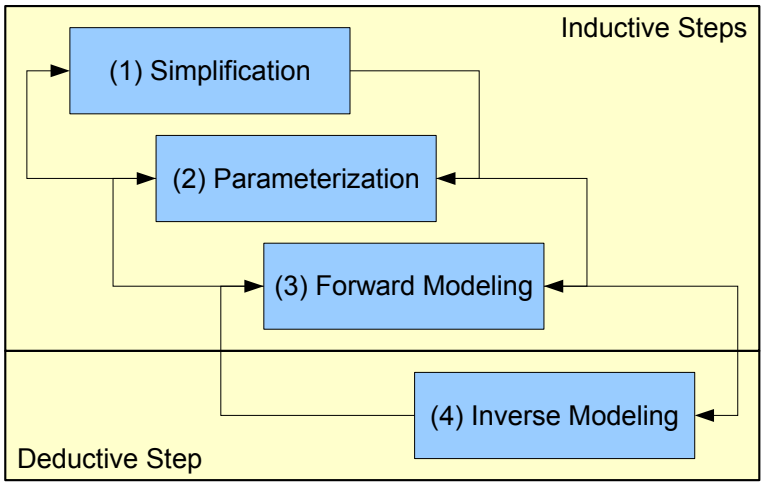


Figure 3.1: System Identification Steps

This step can be split up into two parts: considering influences from supersystems and considering influences from subsystems. Of course, defining which the sub- and supersystems are is also part of the action - and these are subjective steps which depend heavily on experience and knowledge. Thus this step is an inherently *inductive* one

This part of system identification will not be discussed further in this review.

3.1.2 Step (2): Quantification

In the previous step we have tailored the system to our needs. Now we have to identify those quantities that we need to fully describe the system. The values for these identified input variables, constants and parameters form the basis for system output predictions of our model.

The choice of quantities is generally not unique. It is up to the researcher if he wants to use, for example, logarithmic values for a quantity or absolute values. Thus it is also a subjective, inductive step. Splitting up the quantities in input variables, output variables and parameters normally is a natural choice.

This part of system identification will not be discussed further in this review. There is an extensive discussion on parametrisation (as he calls it) in [Tar04, p. 2ff, p. 159ff].

3.1.3 Step (3): Forward Modelling

There are different approaches to solve the forward modelling problem. The choice depends heavily on the intended usage of the model and on the available data. This step is discussed in more detail in chapter 4.

3.1.4 Step (4): Inverse Modelling

The step of inverse modelling takes the model created by the previous step and tries to identify missing values for parameters for it by comparing its input and output with measurements of the real system.

Inverse modelling today is performed with the help of a computer using some kind of method. These methodologies are the central focus of this literature review and discussed in chapters 5ff.

3.2 Other Process Organisations

[Ise92] splits up the process of system identification into two parts with several steps and a strong connection between them: "Theoretische Analyse" (theoretical analysis) and "Experimentelle Analyse" (experimental analysis), which correspond roughly to steps (3) and (4) of [Tar04]. [Lju06] divides the process into three different steps and puts them into the "system identification loop" [Lju06, p. 15]. In this loop he considers steps (3) and (4) but adds *experimental design* and *model validation* to the set of steps. [Nel01, p. 7] has a similar structure, with full iteration loops starting from the last step "model validation". But these steps basically form a more fine grained version of the process described here. [FL97] splits it up into two generic parts: "model structure selection" and "parameter estimation", which roughly correspond to steps (3) and (4) in our division. [Esp05] also sees experimental design as an integral part of system identification - they call it "choosing the regressors".

To sum up, we can say that most authors do not employ the first two steps I mention explicitly. Some authors also see experimental design and model validation as part of system identification. In my opinion they are quite right to do so, but due to space constraints in this review I will not deal with these topics and keep the focus on inverse modeling.

Remark 3.1. Thinking in processes like it is done here has quite a strong stand in the software engineering world. There, several different, quite sophisticated processes to create software have been developed. They are able to handle the complexity and problems that arise throughout the process and even improve the processes themselves. It would be interesting to apply the results of software engineering more rigorously to system identification, and to create some kind of "standard process" which one can follow to create a reasonably good mathematical model for a certain system. See [Bal00] for an overview of the processes used in software engineering.

Chapter 4

Forward Modelling

Forward modelling is the initial creation or selection of a mathematical model for the system under investigation. The choice of parameters and variables happened before (section 3.1.2), their actual values will be determined in a further step (chapter 5). In this step we formulate the connections that the parameters have.

This can be done in two quite different but nonetheless complementary ways: we can derive the model from first principles or we can use a general purpose model and adopt it to our system at hand. Of course, also combinations of these two are possible. Thus we end up with essentially three different types of models, each corresponding to one of the aforementioned approaches: white box, black box and grey box models. In this chapter I will discuss their properties with respect to my main topic, the inverse modelling problem, and present some examples.

4.1 White Box Models

White box models are derived from first principles in the respective scientific fields, with only a relatively small amount of parameters left. These parameters have an interpretation in the first principles used to derive the model. An example is the porosity percentage or the permeability value associated with a certain grid cell in an oil reservoir model.

In this review white box models are models in which prior knowledge *dominates* the model but still some parameters are left for the inverse modelling step. This view is different from "true" white box models where *all* internal workings are known in before and thus the models do not depend *at all* on data. This quite orthodox view would invalidate the inverse modelling step – which is a central part of system identification.

Examples for white box models (in my sense) are all models which are derived from (partial) differential equations. They describe a prior known connection between some quantities of a model.

4.1.1 State Space Models

An important special case arises when a *time dependent* system is modelled using a white box model. When the model is discretised to be tractable for a computer, time steps are introduced. This results in white box models in which carry an internal state. This internal state (and thus the output) depends on the state of previous time steps and the parameters. If additionally the input is not observed, the output of such a model is called a *time series* (cf. section 1.1.2.3).

In many cases the additional assumption can be made that the state of the current time step depends only on the previous time step and the parameters. This results in a so-called Markov process model (see also section 2.1.3). Then the so-called *Hidden Markov Model* (HMM, see figure 4.1) or *Gauss-Markov Discrete Time Model* (as [AM79] calls it) can be applied which describes the evolution of a system *without memory* in which the state is indirectly observable through output variables. Generally this setting is described by equations 2.6 - 2.8.

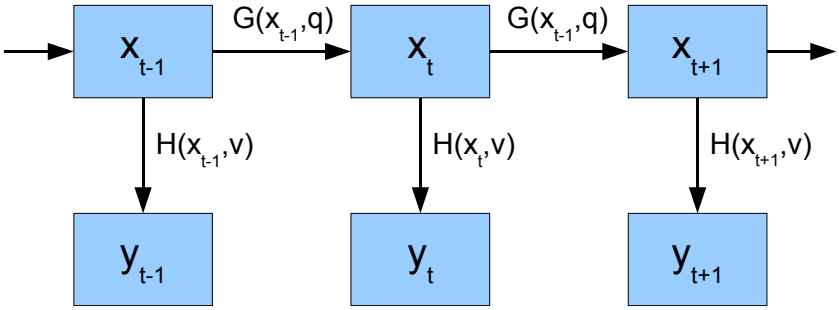


Figure 4.1: The Hidden Markov Model (HMM)

For the linear case with additive, gaussian noise such a model has been described in [AM79, p. 13]:

$$\mathbf{s}_{k+1} = F_k \mathbf{s}_k + G_k \mathbf{w}_k \tag{4.1}$$
$$\mathbf{z}_k = H_k \mathbf{s}_k + \mathbf{v}_k \tag{4.2}$$

Here F_k, G_k, H_k are linear operators and $\mathbf{w}_k, \mathbf{v}_k$ are independent, zero mean, gaussian white noise processes[1]. This can be readily generalized to non-Gaussian noise, see for example [Kit87]. The other direction is the generalization to non-linear operators [AM79, p. 194][2]:

$$\mathbf{s}_{k+1} = f_k(\mathbf{s}_k) + g_k(\mathbf{s}_k)\mathbf{w}_k \tag{4.3}$$
$$\mathbf{z}_k = h_k(\mathbf{s}_k) + \mathbf{v}_k. \tag{4.4}$$

The class of state space models describes the time structure and almost always also the error structure of a dynamic process. Often measurement errors and model errors are modeled and can then be treated by the estimation methods. The physical relations, on the other hand, are not touched and are generally modeled with a white box model – which is a quite useful property of these models. [Nel01, p. 505] puts it this way:

> The major advantage of a state space representation is that prior knowledge from first principles can be incorporated [...].

On the other hand, if a time dependent process cannot be modelled with a white box model because of lack of knowledge of the underlying process the field of *time series analysis* and *time series models* may be useful. Time series models are a special form of dynamic black box models and described in section 4.2.2.

4.2 Black Box Models

In black box modelling we choose a generic model structure or model family containing many parameters. These we have to determine in the inverse modelling step by taking measurements into account.

A property of this type of modelling is that the model parameters have no physical, biological or otherwise logical interpretation. We cannot verify them directly - this is possible only through model validation. We cannot use their values for further reasoning in the target domain. But we can develop inverse modelling algorithms that fit on the model structure and exploit the internals of the generic model.

[1] For this model the famous Kalman filter [Kal60] is an optimal estimator in the minimum mean square error sense (cf. section 8.1.3.1).

[2] In this case, for example the extended Kalman filter [AM79, p. 195] can be used but naturally this filter is no longer optimal like the Kalman filter in the linear case (cf. section 8.1.3.2).

4.2.1 Static Black Box Models

Static black box models can be used to model time independent systems of various types. I will now mention the most popular models, sorted approximately by complexity.

4.2.1.1 Simple Models: Linear Equations, Polynomial Equations and Look-Up Tables

The simplest static black box model is a set of linear equations. This model approximates the relation between input and output by a hyperplane in the n-dimensional space. The parameters can be determined for example by a least squares fit. A discussion of its properties can be found at [Nel01, p. 220f]. A straightforward extension is to use polynomials instead of linear functions. Their properties are discussed at [Nel01, p. 223f].

A different type of non-linear static models are look-up tables. They are the most common type used for low dimensional input spaces because of their low complexity, both conceptual and computational. They are discussed intensively at [Nel01, p. 224-237].

These rather simple models are used often when model performance is of high importance: they can be evaluated quite fast, especially the look-up tables. [Nel01, p. 451] explicitly suggests to try linear models in any case first and only move to more complicated models if the performance or accuracy is not satisfactory.

4.2.1.2 Complex Models: Neural Networks, Fuzzy and Neuro-Fuzzy Models

Artificial neural networks resemble the structure of biological neural tissue. The motivation is that this structure is very well suited for parallel, adaptive information processing. The two most important flavours of neural networks are the *multilayer perceptron* (MLP) *networks* and the *radial basis function* (RBF) *networks*. The parameters in a neural network are generally computed by an optimisation algorithm which usually exploits so-called *backpropagation* to compute derivatives of the output with respect to the internal parameters. This speeds up the optimisation process considerably, of course. Neural networks are discussed in [Nel01, chap. 11].

Neuro-Fuzzy models essentially are fuzzy models which are (at least partly) learned from data and not designed by expert knowledge only. The two most important flavours are *singleton neuro-fuzzy models* and *local linear (Takagi-Sugeno) neuro-fuzzy models*. See [Nel01, chap. 12-14] for a detailed description.

These more complicated models can be used for a huge class of different problems. For example the MLPs are especially suited for high dimensional systems, while RBF networks are better for low and medium dimensions. The latter ones can be reliably trained with linear optimisation techniques, whereas the MLPs require nonlinear training methods. Interesting about neuro-fuzzy models is that it is (to a certain extend) possible to interpret the model with respect to the system, and also incorporate prior knowledge.

4.2.2 Dynamic Black Box Models

In contrast to static models the following models are also capable of handling time dependency of the system. In some cases it is helpful to see these models in the context of state space models (see section 4.1.1).

4.2.2.1 Linear Models

A considerable amount of linear dynamic models exist. They can be cast into the general form

$$y(k) = \frac{B(q)}{F(q)A(q)}u(k) + \frac{C(q)}{D(q)A(q)}v(k) \tag{4.5}$$

or respectively

$$A(q)y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}v(k) \tag{4.6}$$

where k is the time index, q is the forward shift operator[3], u is deterministic input and v is stochastic input (white noise). The process of system identification then has to determine values for the parameters of the polynomials A, B, C, D, F. The notation is according to [Nel01, Lju06].

Depending on which parts of the equation are different from 1 (for the polynomials) or 0 (for the input and noise) the resulting models have different names. For example,

$$A(q)y(k) = B(q)u(k) + v(k) \tag{4.7}$$

is the famous *autoregressive with exogeneous input* or ARX model, where we have $F(q) = C(q) = D(q) = 1$. In contrast the model of the form

$$y(k) = B(q)u(k) + v(k) \tag{4.8}$$

is the *finite impulse response* or FIR model.

Further information on this type of models is available, for example, in [Nel01, chap. 16].

4.2.2.2 Non-Linear Models

There are several approaches to non-linear dynamic systems. The most general one is to replace the linear relationship in equation 4.5 with an arbitrary one, which we will call h:

$$y(k) = h(u(k), u(k-1), \dots, v(k), v(k-1), \dots, y(k-1), y(k-2), \dots) \tag{4.9}$$

Then the problem of determining the linear parameters of this equation is extended to determining the non-linear function that relates the input to the output. According to [Nel01, p. 548] these are called *external dynamics* models.

A more general framework is given by state space approaches where we have access to previous timesteps only through a (possibly non-linear and noisy) measurement operator g:

$$m(k) = g(y(k), w(k)) \tag{4.10}$$

Here w is a noise process acting on the measurement operator g and m is the result of the measurement operator when measuring y (usually simply called *measurement*).

If this measurement operator is not able to deliver a complete set of measurements of all states we arrive at a class of models with internal states, which [Nel01, p. 549 and chap. 17.3] calls *internal dynamics* models. These models are hard to identify as we have to identify the model h, the measurement operator g and the internal states of the model to obtain a correct model.

Polynomial approaches to non-linear system identification now approximate the non-linear mapping h by a polynomial of a certain degree and we are left with the task of determining its coefficients [Nel01, chap. 18].

Another approach is to use neural networks and fuzzy models for the external dynamics approach described above [Nel01, chap. 19,20], but also using neural networks with internal dynamics is possible [Nel01, chap. 21].

4.3 Grey Box Models

Grey box models are, just as the name suggests, a combination of white box and black box models. The amount of first principles used is in balance with the amount of experimental data [Nel01, p. 16]. Grey box modelling is also known as *semi-physical modelling* [FL97].

[3] The notation using polynomials in the *forward shift operator* q is created to be consistent with the conventional notation of the z-transform. See also [Lju06, p. 81].

4.4 Literature Summary on Model Types

[Jat06, p. 5–6] divides the model types into "phenomenological models" and "behavioural models". This essentially corresponds to white box and black box models. He further divides the phenomenological models into "parametric" and "non-parametric" models. The first kind are models like "state space models" and "transfer function models", the second kind "impulse- or step-response models", "frequency response models" or "power spectral density models". Most other literature uses the same distinction between white box, black box and grey box models as presented in this review.

4.5 Discussion

White box modelling is more time consuming than black box modelling, at least in the forward modelling step. We need detailed domain knowledge to design the model. But it provides good understanding of the true system and is reliable. Such models are often used for *extrapolation* purposes. They can also be used for planning, construction and design purposes, as we can sometimes even build them without having any measurements of the true system.

Black box models, on the other hand, tend to be easily derived, even without explicit domain knowledge, by simply incorporating the measurements into a generic model structure. A big advantage is that we can use these models even if we do not have any understanding of the true system. Drawbacks are that we cannot use them for reliable extrapolation. Their accuracy is limited by the measurement errors. As they are based heavily on measurements we can use them only for existing systems [Nel01].

4.6 Choosing a Model Type

Important criteria for choosing a model are:

- For which application is the model intended?
- Which information sources are available?
- Which features do we need and which drawbacks do we allow?

In this review I use the model type distinction from [Nel01, p. 15–16]. It seems logical that in system identification (which deals with "true systems" and "measurements", after all) there are only shades between the extreme model types "white box" and "black box", and true white box or black box models are seldom used.

4.7 Further Reading

In this review I only scratch the parts of forward modelling which are relevant for the inverse modelling step. There is of course much more to this topic. See [Sil01] for a nice article on "modelling as a discipline" with some anecdotes on improper modelling. [Nel01] contains much information on linear/non-linear static/dynamic black box modelling. An attempt to structure many probabilistic models into a unifying framework called "Bayesian programming" is made by [DBM03].

Chapter 5

Inverse Modelling

In the inverse modelling we finally determine the actual model parameter values which were left over from the forward modelling step by taking into account measurements of the true system, involved uncertainties and modelling issues.

5.1 Methods Corresponding to Views

The methods which are available depend on the view that one takes on the problem as described in chapter 2:

- The **probabilistic view** leads to methods which update the "state of information" [Tar04, p. 6] one has on the problem. These are often Monte Carlo methods which aim at solving the update equation 2.3 on page 12 by stochastic sampling. But special assumptions on the involved uncertainties lead to deterministic methods (which also make the connection to least squares (optimisation view) and Kalman filter (state space view) methods).
- The **ill-posed optimisation problem view** leads to so-called regularisation methods which help to minimise the ill-posed inverse problem of equation 2.5 on page 13. Basically they introduce additional knowledge one has on the solution into the equation and by this make it "regular" (hence the name). Then the inverse problem can be solved by standard optimisation procedures.
- The **state space view** naturally leads to sequential methods, as the view itself is centered around the sequential nature of the problem.

These three different groups of methods are not clearly separable and have some common features. But for the sake of a structured approach I will try to sort them into three groups:

Probabilistic Methods are, naturally, mainly employed in the probabilistic view. They do not genuinely employ an optimality argument in their derivation, mainly as they try to solve the fully probabilistic update equation – and by this theoretically compute the full solution to the inverse problem as defined by [Tar04].

Optimisation Methods are the combined group of regularisation and optimisation methods. They primarily try to find a *single*, somehow *best* solution to the inverse problem, possibly with some attached error estimation. Note that there are stochastic optimisation methods, but in contrast to above methods they only compute a single, somehow optimal solution, too.

Sequential Methods are the methods associated with the state space view on the problem. Historically they are deterministic optimal filters, but over the last years and the increase of computing power stochastic filters (so-called sequential Monte Carlo methods) gained much importance.

5.2 The Following Chapters

In the following three chapters I present the most important types of algorithms for each view. The main separation I use there is between *deterministic* and *stochastic* methods, which essentially come up in all three views. Examples of concrete algorithms and literature references accompany them.

5.3 Stochastic Methods

Stochastic methods are generally known as "Monte Carlo methods". The name was already coined by the seminal paper from Nicholas Metropolis and Stanley Ulam [MU49]. They describe it as originating from statistical mechanics where one is not considering individual particles but sets of particles. And that sets of particles have also been used to approximate continuum physics.

Monte Carlo methods employ (pseudo-) random numbers to conduct their *experiments*. Their aim is to compute the solution of a problem as a statistical approximation derived from this large numbers of experiments. The original method has been modified and optimised in several ways since its first appearance, resulting in a whole family of Monte Carlo methods. Those which are applicable to the problem of inverse modeling I will mention in the respective section of the next three chapters.

5.3.1 Monte Carlo Methods: Example

An illustrative example is the approximation of π by means of a Monte Carlo method: draw as many uniformly distributed random tuples $(x_1, x_2) \in [0,1]^2 \subset \mathbb{R}^2$ as you can. If N is the total amount of experiments and M is the amount of experiments that results in tuples which lie within the quater of the unit circle which corresponds to $[0,1]^2$, then an approximation for π is given by $4 \cdot N/M$.

5.3.2 Simple Monte Carlo Methods

Above example illustrates how simple randomised algorithms can work: they do a large amount of *indepenent* experiments and compute statistical values from the outcomes. There are many examples of such methods, for example the Miller-Rabin-Test for primality or above method to approximate π. Numerically evaluating integrals is also possible by evaluating the integrand at uniformly distributed, random positions within the integration limits. The simple Monte Carlo approximation for the value of the integral is the expected value computed from all experiment outcomes. It can be shown that this value converges to the true value of the integral *almost surely* as the number of experiments goes to infinity.

The methods used in inverse modelling are often much more sophisticated to speed up the solution process, but the central idea of conducting many experiments is still there.

Chapter 6

Probabilistic Methods

Probabilistic methods essentially discretise equation 2.3 on page 12 by sampling from the posterior distribution $\sigma_M(\mathbf{m}, \mathbf{x})$ using a certain sampling procedure. In some cases this "sampling from the posterior" can be done in a deterministic way[1], using additional assumptions. But most of the time, additional assumptions do not hold and we have to revert to more general methods. Thus the probabilistic mainly relies on stochastic sampling procedures. Their main difference to stochastic optimisation methods described in section 7.4 on page 32 is that the results have a clear statistical meaning and we can compute statistical moments from them. In optimisation we can only search for a single solution which possibly has some probabilistic interpretation, like maximum likelihood point etc.

With stochastic filters, described in section 8.2 on page 39, things are different: most of them have a probabilistic interpretation. But their origin is different, so we treat them independently.

6.1 Deterministic Probabilistic Methods

Deterministic probabilistic methods solve the "state of information" update equation using deterministic methods. This needs assumptions on the involved uncertainties. The most important and best known example arises when all involved uncertainties are Gaussian and the model function G is linear. Then the posterior distribution is also Gaussian and it can be fully represented by a mean value and a covariance matrix. The more non-linear the model G is, the further away the posterior distribution is from being Gaussian - but in some cases we can live with that error and use the methods nonetheless. An intuitive idea about "how far we can go" with this is given by figure 3.2 of [Tar04, p. 65].

In the case of Gaussian (or at least sufficiently similar) uncertainties, the mean and covariance can be computed using standard optimisation methods, as presented in [Tar04, p. 62ff]. So I will not present any concrete methods here and point the reader to chapter 7 on page 29.

6.2 Stochastic Probabilistic Methods

Stochastic probabilistic methods are called *sampling* methods. They are exactly what we need to solve equation 2.3 on page 12: they are able to create random samples from the posterior distribution. They can be used to sample from arbitrary posterior distributions, but the main drawback is: they are often horribly slow from a computational point of view. Especially when the likelihood function cannot be simplified as in equation 2.4 on page 13, we have to do at least one integration over the input variable space for each sample we get – and often a lot more.

[1] This is not called sampling anymore, of course. It often results in a least squares problem, which is solved with the help of optimisation methods.

6.2.0.1 Inversion Method

The inversion method is simple and efficient, but not often applicable. For a given pdf f, if we are able to give its cumulative distribution function (cdf)

$$F(x) = \int_{-\infty}^{x} f(x')dx' \tag{6.1}$$

and invert it, namely give an explicit formula for F^{-1}, we can create samples from f by sampling from the uniform distribution $U(0,1)$ and "putting them through" the inverse cdf. It is easy to show that this method is correct, e.g. [Tar04, p. 48].

6.2.0.2 Acceptance-Rejection Method

Acceptance-Rejection (A-R) method (sometimes only called "rejection method") creates samples from a target density using a proposal density (often called "blanketing function") from which samples can be simulated. The algorithm was used by John von Neumann, but can be dated back to "Buffon's needle".

It essentially simulates a sample from the blanketing function and then tests if it is rejected according to a simple rule. If it is not, it represents a sample from the target density. This method works for low dimensions. For higher dimensions the probability of a rejection is too large and it takes too much time to create a reasonably sized set of samples from the target pdf. See for example [CG95] for a short overview.

6.2.0.3 Metropolis-Hastings Sampling

More sophisticated Monte Carlo methods are sequential sampling methods. The most popular sampling methods are Metropolis-Hastings [MRR+53, Has70] (M-H) and a special case of it, the Gibbs-Sampler [GG84]. They are Markov Chain Monte Carlo methods which sequentially sample from an unknown target distribution using a proposal distribution – so they are similar to A-R sampling. The main difference is that the experiments they conduct are not independent from each other - they form a Markov chain, often called a "random walk" in this context.

For the proposal distribution we have to be able to calculate the probability to *move to another state given a current state* (and back, if it is not symmetric). For the target distribution we only have to be able to calculate its probability. An important advantage of M-H-sampling is that we do not have to take the normalising constant k into account. A self contained, well written introduction to the Metropolis-Hastings algorithm can be found in [CG95].

There are also extensions and improvements to this basic algorithm: for example the "multiple-try Metropolis" sampler proposed by [LLW00]. It involves local optimisation steps into the M-H-sampler. Accoding to them numerical studies show that it performs significantly better than the traditional M-H-sampler.

6.2.1 Simulated Annealing

Simulated annealing (SA) is a method designed to obtain the maximum likelihood point of the posterior probability density. It is an optimisation algorithm, but it uses a M-H-sampler at its core to obtain this point, so we can also count it to the stochastic probabilistic methods – the probabilistic interpretation of the result is that it is the maximum likelihood point. [Tar04, p. 54f] quickly points out the connection to the M-H-sampler and gives a simple example. See also section 7.4.3 on page 33 for more information on SA.

Chapter 7

Optimisation Methods

For the optimisation problem view the first important step is to formulate the problem in a correct, consistent and well-posed way. As we are dealing with ill-posed optimisation problems we have to put special attention to the well-posedness.

7.1 Formulation of the Optimisation Problem

To formulate the optimisation problem we have to think about some questions.

What is the criterion for a "good" solution? This is often a straightforward question to answer: taking the difference between measured data and the output of the model (the error) and measuring it in some norm is the most common one. The criterion for a good solution is called "objective function" in the context of optimisation.

But sometimes several concurring answers exist and have to be treated. We then have to decide whether we want to combine them in advance and treat them as one target or if we want to consider all of them and perform a trade-off between them later. The latter approach requires us to use so-called multi-criterion or multi-objective optimisation methods.

In this review the cost function involved in *variational methods* is seen as a special type of objective function[1]. It is often a weighted least squares formulation which is minimised using standard algorithms described in this chapter. Thus variational methods are also treated in this chapter (section 7.5 on page 34).

How do we want to treat outliers in the data? There are different ways to measure the error between the output of the true system and the output of the mathematical model. The most famous is the family of l_p-norms:

- $p = 1$ results in "sum of absolute residuals". It does not pay much attention on outliers.
- $p = 2$ results in "least squares". The weight on outliers increases quadratically.
- $p = \infty$ results in "minimax approximation" or "Chebyshev approximation". It puts infinitely much weight on the outliers as they alone determine the norm.

Other treatments are possible, for example weighted norm approximation or even a generic penalty function approximation [BV04, p. 291ff].

[1] This is also done elsewhere, e.g. [PJHJH07, section 3].

Are there constraints on the parameters? If there are constraints put on the parameters which cannot be eliminated by a change of variables we have to use special optimisation methods to treat them.

Do we have to improve the condition of the problem? To be able to solve an ill-posed optimisation problem satisfactorily it is often necessary to introduce additional information on the solution. This process is known as "regularization" and discussed in section 7.6.1.

7.2 Choosing an Optimisation Method

The choice of method is heavily dependent on the formulation of the optimisation problem and the type of model used. But there are also additional aspects which I will discuss now.

Is it a convex or non-convex problem? In most literature there is a differentiation between linearity and non-linearity of an equation, also for optimisation problems. But in the last few years it became clear that the more important distinction is between convex and non-convex problems [BV04].

Convex problems can be solved almost as efficient as linear ones and even with very similar methods as they have a single global solution. Non-convex problems can have several local minima and many optimisation algorithms tend to get stuck inside one of them (depending on the starting value), although there are methods that try to mitigate this (tunnelling methods).

Do we need a locally or globally optimal solution? The answer to this question is tightly coupled with the previous one: it only makes a difference for non-convex problems. And there it makes a huge difference: identifying the global minimum for a non-convex problem is considerably more complicated and often virtually impossible.

The strength of stochastic optimisation methods (or meta-heuristics) is this global optimisation. They are able to "escape" local minima and move on with their search.

Are the model parameters discrete, continuous or mixed? If we have discrete or mixed model parameters the optimisation problem becomes more complicated: it is a so-called combinatorial optimisation problem and special algorithms have to be used.

To which degree do we need any uncertainty quantification? If your answer is "not at all" you're probably wrong: in system identification we rely on measurement data. And measurements are inherently inexact; they always contain a "measurement error". This is why reasonable sensors come with a "uncertainty quantification". And this information we have to take into account when determining model parameters from noisy data.

This can be done by trying to obtain a quantification for the uncertainty in the model parameters based on guesses for the model errors and the measurement errors. Sometimes it is possible to use *robust optimisation* (e.g. [GCC08]) approaches which incorporate a safety margin into their results.

Are continuous derivatives of the error- or cost-function available? If we have derivatives available we can use first, second or even higher order methods for optimisation. This will considerably speed up the process - but generally these methods cannot be used for global optimisation as they are only able to find the next local optimum, dependent on the start point of the search.

Can we exploit the model structure in some way for the optimisation algorithm? This can be a difficult question to answer. It is especially interesting for grey box and white box models, as black box models normally come together with specialised training algorithms (which is nothing else than the optimisation step).

For example, by taking advantage of the model structure we can reduce the computing time required to obtain results of a certain accuracy. In any case exploiting the model structure means adapting an optimisation method for a certain problem by hand, or even designing a combination of several optimisation methods that fits the problem. This can obviously be a hard, lengthy task.

7.3 Deterministic Optimisation Methods

Deterministic optimisation methods can be grouped by several properties. We will use the type of objective function they are applicable to as a differentiation criterion.

7.3.1 Discrete Methods

Discrete methods are used to find (mixed) integer solutions to linear optimisation problems. Examples of such methods are the *cutting-plane method* [Gom58, Gom02], *branch and bound* [LD60, Dak65], and a combination of the previous two, called *branch and cut*.

7.3.2 Direct Methods

Direct deterministic methods find solutions to continuous optimisation problems by directly evaluating the objective function at certain points. No gradient information is used, so the objective function is not required to be differentiable.

The simplest example are so-called *relaxation methods* or *univariate methods*: they optimise one variable at a time and search for each the optimal value [Spa62]. These methods are simple to understand and implement, but not very efficient.

Popular examples of more sophisticated methods are the *(Danzig) simplex method* for linear programs from 1947, for which no real seminal reference can be given [Dan02]. An extension to non-linear objective functions is the *Nelder-Mead simplex* or *downhill simplex*[NM65]. On the way to modern convex optimisation lies the *ellipsoid method* or *Nemirovsky-Yudin-Shor method* independently proposed by Nemirovsky and Yudin (and a special case of something which Shor developed) in 1976-77. It was proven to solve linear programs in polynomial time by Leonid Khachiyan in 1979 [Kha79], which was a major step towards modern optimisation methods. In 1984 Narendra Karmarkar introduced the first of the famous *interior point methods* [Kar84], then also for linear programming only. Later Yurii Nesterov and Arkadii Nemirovskii showed how to use these methods for convex optimisation in general [NN94].

7.3.3 First-Order Methods

First order methods not only use the objective function directly but also gradient evaluations of it. This gradient information (or at least an approximation to it) has to be available and accessible for these methods to work.

Univariate methods can also be combined with gradient information: they use the variable in which the gradient has largest absolute value [Spa62]. But again, these methods are not very efficient.

More sophisticated first order methods are *gradient descent methods* or *methods of steepest descent*. They are iterative optimisation methods which uses the negative gradient information at the current point and go into that direction with a certain step size. It is often combined with a *line search* to determine the best step length at each iteration. Simple steepest descent was known at least as early as Cauchy [Cau47]. A more recent example is [FP63].

A problem of these methods is that successive optimisation steps may be taken in the same direction. This is especially a problem with linear optimisation and is mitigated by methods of conjugate directions, e.g. the *(preconditioned) conjugate gradient method* [HS52]. A non-linear conjugate gradient method is also available [FR64] but due to the non-linearity the orthogonalisation step has to be re-done every few iterations [FR64, Pow77]. For linear problems this can also be the case due to numerical problems, although there are modified conjugate direction methods which try to mitigate this problem (e.g. [Yuk07]).

7.3.4 Second-Order Methods

These methods make use of the Jacobian of the objective function. The most popular one is of course *Newton's method*[2], sometimes also called *Newton-Raphson method*. It can also be modified to achieve better convergence for certain special cases. Then it is called a *modified Newton method*.

Quasi-Newton or *variable metric methods* replace the explicit evaluation of the Jacobian by an approximation, so they can be regarded to lie somewhere between first and second order methods. The most popular variable metric method today is *BFGS* [Bro70a, Bro70b, Fle70, Gol70, Sha70] and a variant for large-scale problems called *limited memory* or *L-BFGS* [Noc80, LN89]. This was developed because the explicit formation of the approximate inverse of the Jacobian is generally a dense matrix, which is too large for large-scale problems. In L-BFGS this matrix is replaced by storing only the last n update vectors to this matrix.

7.3.5 Higher-Order Methods

There are also higher order methods, especially the family of *Householder's root finding methods* [Hou70]. They use higher order gradient information, so Newton's method is the Householder method of order one. The next one, which uses Hessian information, is *Halley's method*. But these methods are rarely used, as the expense of evaluating higher order gradients is not worth the improved convergence rates.

7.4 Stochastic Optimisation Methods

Stochastic optimisation methods are often heuristics derived from natural processes, like biological or physical ones. In these methods, (pseudo-) random numbers play an important role, so they can also be considered as extensions of the original Monte Carlo method (see section 5.3 on page 26), just like the methods described in section 6.2 on page 27. But the methods described below generally do not have a probabilistic interpretation.

7.4.1 Random Search

While being probably the most inefficient optimisation method of all, it is also probably the easiest one: pick a random point in search space and evaluate the objective function. This can be efficiently parallelised, of course. When for some time no improvement could be made, the method has "converged".

All of the following stochastic methods somehow incorporate aspects from this simple idea. They improve the convergence speed by introducing iterative structures and other clever ways to steer the random seach towards optima. Many of these methods are heuristics, "rules of thumb", as they have no mathematically rigorously provable structure, like, for example, the Newton's method or BFGS. Nonetheless they have shown to come to good, applicable results.

[2] Originally Newton's method is a root-finding method, e.g. it solves $f(x) = 0$ for x. But knowing that for $f(x) = \min$ we have $f(x)' = 0$ we can also use it for numerical optimisation.

7.4.2 Biologically Motivated Methods

Methods derived from living beings are probably the most popular heuristics used in optimisation. They use a population of solutions which is iteratively improved using some strategy.

7.4.2.1 Evolution Biology

From evolution biology two main streams of methods have been derived: genetic algorithms (GA) [Gol89] and evolution strategies (ES) [Sch75]. They both use the analogy of a population of individuals where the fit (according to the objective) ones are allowed to reprodcue with a higher probability than the less fit individuals. Reproduction is performed by recombining two (or even more) individuals to a new one, which is expected to perform better with a good probability. Additionally random mutations can occur.

Both methods differ essentially in the way they represent the individuals: GAs encode the solution values into binary strings. All operations of the GA, like recombination and mutation, are then performed on these genotypes. This resembles the actual biological process: the individuals, called phenotypes, are evaluated in their environment during life. But the recombination is performed on the genotype. This allows, for example, to encode mixed-integer problems like the travelling salesman problem and solve them using GAs.

ES are a little bit different: the operations are performed directly on the decision variables, which are expected to be a vector of real numbers. The difference between genotype and phenotype is smaller: the genotype may contain additonal parameters of the algorithm which are optimised along with the decision variables themself. This results in a self-adaption process, which is a big advantage of this type of algorithms.

GAs and ES are also well-suited for multi-objective optimisation, for example see [ZLT01] and [DPAM02, LZ07].

7.4.2.2 Swarm / Group Behaviour

Another class of biologically inspired optimisation methods is derived from swarm or group behaviour. These methods also employ a population of individuals, but they are not "dying and reproducing" like in section 7.4.2.1.

For example in the "particle swarm optimisation" methods the individuals have a certain "speed" in the solution space and try to mimic the behaviour of the best performing individual. Of course, also for this algorithm many improvements and extensions are available. A good overview of "particle swarm optimisation" methods is given by [BVA07, BVA08].

Further examples include the bee's algorithm [PGK+06] or ant colony optimisation [DS04]. More "obscure" ideas of this area are harmony search [GKL01], which mimicks the improvisation of music players, and an algorithm mimicking bacterial foraging [Pas02].

7.4.3 Physically Motivated Methods

The most popular optimisation method derived from physical processes is *simulated annealing*, introduced by [KGV83] in 1983. It mimics the annealing of matter from high temperature. On its way the material is seeking a state of minimal energy. This behaviour can be used as a global stochastic optimisation method: the state of minimal energy is the global solution to the optimisation problem. But there is also a connection to probabilistic methods: at its root the simulated annealing method has a Metropolis algorithm, 6.2.1 on page 28. but as its original idea is to find a maximum likelihood point we consider it more as an optimisation method.

There is a multitude of additional and modified methods avaiable, which try to circumvent problems of original algorithms and improve the convergence, either generally of for specific problems. Thus I can only mention a few of them:

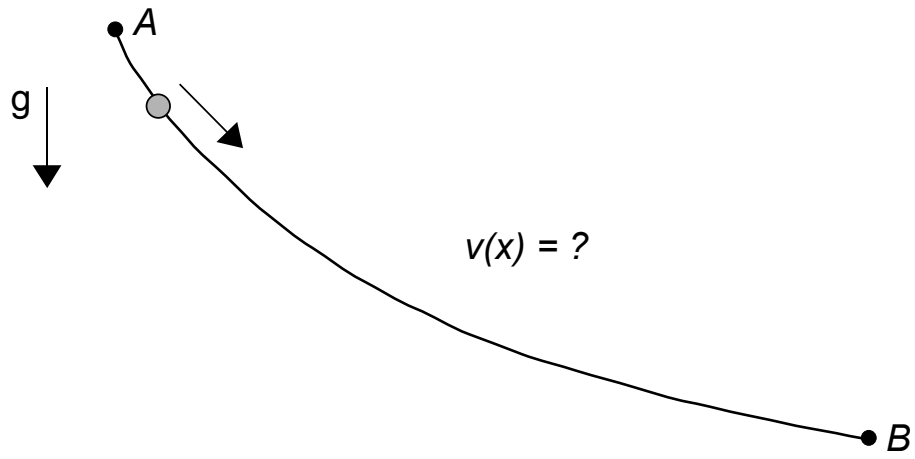


Figure 7.1: The Brachistochrone Problem

The first example is stochastic tunnelling [WH99, Ham06], a generalisation of simulated annealing which tries to circumvent the "freezing problem" by tunneling (see section 7.6.2 on page 36) energetically inaccessible regions.

Another one is quantum annealing, an extension of the classical simulated annealing, is another optimisation method motivated by physical processes. In addition to the temperature in simulated annealing, it uses quantum mechanical characteristics of a pyhsical system. This can be done by variing the masses of the constituent particles. It also has tunneling (see section 7.6.2 on page 36) aspects in it. For details, see [FGS+94] and the references therein.

7.5 Variational Methods

The early roots of variational methods lie in the calculus of variations [JGJS99]. It deals with finding extreme values of a *functional*, so the variables here in general are functions. Thus it is normally an example of infinite dimensional optimisation. An illustrative problem from the history of calculus of variations is the Brachistochrone problem (see figure 7.1):

Problem 7.1. *A bead is threaded between two points of different height with a certain distance in between. The bead can descent on this thread without friction under the influence of gravity. What is the shape of the thread that gives the minimum descent time between these two points?*

The solution can be computed using calculus of variations, e.g. see [Wei]. The variational methods which are in use today have been derived from optimal control theory and objective analysis [NJC06, p. 379].

Variational methods essentially replace a difficult problem by a simpler one with free, so-called variational parameters which must be fit to the problem, e.g. optimised [JGJS99, section 4]. In the context of inverse modeling a (often quadratic) cost functional, the objective function, $\mathcal{J}[\psi]$ is formulated. The first variational derivative[3] of this functional, namely $\delta\mathcal{J} = \mathcal{J}[\psi + \delta\psi] - \mathcal{J}[\psi]$, must approach zero when the arbitrary perturbation $\delta\psi$ goes to zero. This gives rise to the famous Euler–Lagrange equations, which can be somehow regarded as analogous to Fermat's theorem in calculus.

But this also means that we have to compute variations of the searched-for variables – which in inverse modelling essentially means the computation of *adjoint* models. This is normally manual work and can range from "time consuming" to "impossible", e.g. when the model equations are inaccessible or cast into proprietary computer code.

[3] Therefore the name variational methods.

7.5.1 Representer Method

The representer method minimises the misfit between measurements and models in a least squares sense by minimising a cost functional, like all variational methods. Its distinguishing feature is that the minimiser of the cost functional $\mathcal{J}[\psi]$ is expressed as a linear combination of a solution without taking measurements into account (the forward model) and so-called representers which depend on the measurements. For each measurement one representer must exist. This decouples the Euler-Lagrange equations and renders them solvable.

[BD05] explains the representer quite nicely method using a single-phase Darcy flow in porous media as an example. [PJHJH07] applies the representer method to a simple, onedimensional hydrocarbon reservoir model.

7.5.2 Variational Data Assimilation

Especially in meteorology and oceanography, but sometimes also in other scientific fields, the incorporation of measurements into a model is called "data assimilation". The most popular variational methods for this application are 3D-VAR (three dimensional variational data assimilation, e.g. [VK99]) and its extension to the time domain, called 4D-VAR (e.g. [FHH07]). The latter one uses not only the observations available at analysis time but also those from a specified timeframe before analysis time.

Both methods essentially formulate the standard cost function and minimise it using standard optimisation algorithms. The necessary gradient information is often obtained by means of adjoint methods, e.g. [FHH07]. So the 3D-VAR and 4D-VAR are not fixed algorithms which can directly be programmed into a computer. They are classes of methods which follow a common idea but may result in different implementations, according to the problem at hand.

Today, 3D-VAR and 4D-VAR are standard approaches used for weather forecasting. For example the "European Center for Medium Range Weather Forecasts" uses 4D-VAR approach [RJK+00].

7.6 Supporting Methods

These methods are not optimisation methods in their own right, but they are sometimes helpful, sometimes even essentially required, to obtain solutions to the inverse modelling problem.

7.6.1 Regularisation

Regularisation methods introduce additional information into the process of parameter estimation. This information often comes in form of smoothness properties or norm bounds. Regularisation methods help to determine parameters from measurements by desgning an approximate inverse of the model operator. This is necessary as computing the generalized inverse of the model often is an ill-posed problem. Regularization methods can consider noisy data and poorly known model equations as well. See [EHN00] for a comprehensive introduction. [Neu98] also contains much information.

Most theory is developed for linear (infinite dimensional) model operators, although some theory for non-linear operators is also available.

The most popular regularisation method is probably the Tikhonov regularisation, both for the linear and non-linear case. Iterative regularisation methods are the (linear and non-linear) Landweber iteration and the conjugate gradient method. Also well-known is the Levenberg-Marquard method, which can be derived as Tikhonov regularisation applied to a Newton step [EHN00, p. 285].

Of course there are many variants and extensions to these approaches. For example see [LPR07, BK08, DES98, Kal97].

Regularisation methods are popular with image and signal processing, Radon transform inversion (X-Ray tomography), inverse scattering but also parameter identification.

7.6.2 Tunneling

Tunneling [LM85] is an interesting technique to locate global optima. It uses any minimisation method to find a first (local) minimiser. Then it employs a special function to exclude this mimimiser from further search and employs any minimisation method to minimise this "helper" function instead. The result becomes the starting point for the next minimisation of the original target function. The process is performed iteratively.

7.6.3 Homotopy Methods

Homotopy methods can also be seen as part of regularisation. But their generality justifies their separate treatment.

Homotopy methods split up the solution of a problem which is difficult to solve (whatever difficult may mean here) into a sequence of simpler problems. The major trick is to re-use the solution of the previous, easier step as starting point for the next step. This next step in turn becomes easier to solve if the solution of the previous step is "close to" the solution of the current step. This is a certain continuity requirement on the solution. If it is fulfilled the solver has to go only short steps when compared to the full way from the initial guess to the final solution. This is easier to do in an intuitive sense.

This approach can readily be applied to the regularisation parameter (often called α) that arises in most regularisation methods. Such methods are also actively developed, for example see [MKT07].

7.7 Model Structure Optimisation

It is also possible to perform an optimisation on the model structure. This is especially interesting for black box models (see section 4.2 on page 20), where the model itself has opaque parameters like number of hidden layers in a neural network. This type of optimisation will not be considered further in this review.

Chapter 8

Sequential Methods

Sequential methods are methods which sequentially (often called "recursively") update their result with a measurement data stream. These methods use measurements of the real system and the model operator G to approximate the unknown state $\mathbf{s}_t$. This can be done for the state of current timestep $\mathbf{s}_t$ using measurements up to and including it – then the process is called *filtering*. If measurements from future timesteps are also incorporated the process is called *smoothing*, and if measurements of only past timesteps are incorporated the process is called *prediction*. These sequential processes, together with the variational methods described in section 7.5 on page 34, are also called *data assimilation*, especially in meteorology and oceanography.

Sequential methods are often used to make predictions on future states of the system under consideration, for example to predict stock market behaviour or to predict weather. But also making estimations of the current state can be a purpose, as in the radar tracking example. Smoothing can be used to gain more insight into the dynamic behaviour of a system.

8.1 Deterministic Filters

8.1.1 Recursive Least Squares Filter

The recursive least squares (RLS) filter essentially is the recursive equivalent to the classical least squares optimisation algorithm. Its formulation allows for sequential inflow of data (like all filters) into a least squares fitting problem. This makes it usable for online real time applications.

We can think of the least squares algorithm as Newton's method applied to the quadratic loss function of least squares - which converges in each iteration step, as it uses curvature information of the target function (the Hessian) to find its minimum. This curvature information is in the quadratic case perfect, of couse - so we do not have to iterate. We can safely say that "RLS is a recursive version of Newton's method applied to a linear optimization problem" [Nel01, p. 60].

The extension to non-linear target functions is simple: we end up with the "non-linear least squares" problem - which can be solved, for example, by applying Newton's method to it. The resulting algorithm, though, is nothing else than the famous Gauss-Newton method. So in this yet simple but important algorithm we see a direct connection between optimisation algorithms and filter algorithms.

Remembering 6.1 we can also see a connection to probabilistic methods: in the linear case and with additive Gaussian uncertainties the solution is, again, given by solving a least squares problem [Tar04, p. 64ff]. So in this "simple" case all views and the resulting methods are somehow "the same" (though they may differ in their derivation and implementation).

The RLS filter is actively developed, especially to reduce its computational complexity. These so-called *fast* RLS filters are employed in the field of control engineering, where they are used for online system identification [Nel01, p. 64].

8.1.2 Least Mean Squares Filter

The least mean squares (LMS) filter, invented by Bernard Widrow and Ted Hoff in 1960 at the Stanford University, is essentially a recursive version of the method of steepest descent. It converges, of course, slower than RLS as it does not employ curvature information. On the other hand its computational complexity is even lower than RLS, so it makes sense to employ it in real time applications [Nel01, p. 62],[HW03].

8.1.3 Kalman-Type Filters

Starting with the seminal paper of the Hungarian-American mathematician Rudolf Emil Kálmán in 1960 [Kal60] a multitude of similar filters has been developed. They are more or less closely related, especially with the inherent assumption that the errors involved are additive and their pdfs are Gaussian distributed with known mean and standard deviation. In practice this so-called *Gaussian assumption* is relaxed to a certain extent, e.g. the filter is applied to problems where the error is "reasonably well" described using a Gaussian distribution.

The original filter and many of its successors are deterministic algorithms, although there have been also some interesting developments in the field of stochastic filters which use some of the mechanisms involved in the original filter.

8.1.3.1 Linear Kalman Filter

The original Kalman filter is closely related to RLS with forgetting [Nel01, p. 65]. The main difference is that the quantities to be estimated are (dynamic) state variables instead of static parameters. For linear system models and measurement operators and additive Gaussian noise the original Kalman filter can be proven to be the optimal filter.

8.1.3.2 Extended Kalman Filter

The extended Kalman filter is the generalisation of the standard Kalman filter to non-linear target functions. It is suitable if the system model is slightly non-linear, meaning that it can be described by its first order linearization (the Jacobian) sufficiently [AM79]. So essentially the linear transformation within the Kalman filter is replaced by the Jacobian. This approach has several problems:

- The filter is no longer optimal as in the linear case.
- Jacobian information must be reasonably sufficient to describe the non-linearity of the transformation. If it is not it may degrade the filter performance or, in the worst case, lead to filter divergence.
- Jacobian information must be availabe – this is not the case in all applications, e.g. when the transformation function contains discontinuities.
- Calculating the Jacobian matrix is, depending on the complexity of the transformation, subject to derivation and implementation difficulties. These errors might cause unnoticeable problems that lead to erroneous filter behaviour or bad performance - which cannot always be detected.

To give a seminal reference for the extended Kalman filter is somewhat impossible - it seems to have been developed quickly after the original Kalman filter and many authors are cited with it in 1960s and 1970s. But the one who really came up with it is, at least to me, unknown.

8.1.3.3 SEEK Filter

In the singular evolutive extended Kalman filter the updating works in almost the same way as in the standard and extended Kalman filter. The difference is that the update is not allowed to be on the full search space dimension but only on a smaller subspace of it which is tangential to an assumed attractor in this state space. The basis for this subspace is obtained by the famous principal component analysis (PCA), also called empirical orthogonal functions. Details can be found in [PVCR98, Pha97, Pha01].

8.1.3.4 Unscented Kalman Filter

The Unscented Kalman filter was designed to fix a major problem of the extended Kalman filter. It employs the *unscented transformation* (UT) to propagate mean and covariance information (or even higher order information about a distibution) through non-linear operations by choosing so-called sigma points and putting them through the full non-linear operation. The result should reasonably describe the mean and covariance of the resulting distribution [JU97, JU04].

8.1.4 Multimodal Filters

Multimodal filters are often based on maintaining a set of mean and covariance estimates. They can be used, for example, for multiple target tracking. Examples are the sum-of-Gaussian filter [AS72] and multiple hypothesis tracking [Rei79].

8.2 Stochastic Filters

Most filter [results] have a certain probabilistic interpretation, also the stochastic ones. The stochastic filters make use of a set of numerical models that represent these statistics. This set is often called an *ensemble* in the context of stochastic filters.

8.2.1 Ensemble Kalman Filter

The ensemble Kalman filter was developed in 1994 by Geir Evensen, then at the Nansen Environmental and Remote Sensing Center (NERSC) in Bergen, for application with ocean models. It is a Monte Carlo technique similar to the sequential Monte Carlo filters described in 8.2.3 but it uses the assumption that measurement errors and model errors are Gaussian (or, at least, reasonably well described using a Gaussian distribuion). This assumption allows the filter to use a state updating technique similar to the original Kalman filter (section 8.1.3) – in contrast to the particle weighting and re-sampling schemes used in SMC filters (section 8.2.3).

The main difference to the original Kalman filter is that the covariance information which has to be available for the Kalman updating step is derived from the ensemble of realisations instead of keeping and advancing a full, possibly extremely large, covariance matrix. The ensemble members are propagated forward in time using the full non-linear transformation and by doing so the covariance information is updated, too. Originally the filter was developed to mitigate problems with unbounded error growth in the extended Kalman filter when applied to a non-linear ocean circulation model [Eve07, p. 36].

It can also be formulated as an optimisation algorithm: in [Eve07, chapter 10] the Ensemble Kalman Smoother (EnKS), which essentially is an EnKF that propagates information also backwards in time, is shown to solve an optimisation problem in a Bayesian setting.

8.2.1.1 Parameter Estimation with EnKF

As the EnKF is a filtering technique it was developed for updating state variables. In this case the original application was to assimilate data into an ocean model. But as mentioned in [Eve07, p. 123] the state vector can also be augmented with poorly known (static) parameters of the model, which then are updated with the same methodologies as the state variables.

8.2.1.2 Handling Model Errors with EnKF

The filter can also be used with imperfect models. [Eve07, p. 175] shows how to introduce a time dependent stochastic forcing term with a parameter c. And even this parameter can be updated as correlations between measurements and this noise parameter will develop, according to [Eve07, p. 180].

8.2.1.3 Localized Ensemble Kalman Filter

The standard Ensemble Kalman Filter suffers a so-called long-range *spurious correlation* problem [HWS01, AO08], where updates to state and parameters can be observed that are somehow[1] far away from the source of information. According to [HM98] this is due to the small ensemble size: larger ensembles show that these long-range correlations actually do not exist. But for computational reasons we cannot simply choose such a large ensemble size, so different solutions have to be found.

The solution of [HM01, HWS01] is to create a localization matrix that depends on the spatial distance of state variables, parameters and measurements. This boils down to creating a matrix which contains the Euclidean distances of, for example, all relevant grid blocks of a 3D grid. The resulting distances are then put into a function with compact support, e.g. a linear function, that maps the distances to values between 1.0 (for the distance of a grid block to itself) and 0.0 (for the distance of two grid block which are far away from each other). The resulting covariance localization matrix is applied within the EnKF update equation using a *Hadamard* or *Schur product*.

There are also approaches which use other metrics than spatial distances for the computation of the localization matrix. For example [AN06, p. 65f] uses a stream line based metric to compute the localization matrix for an application with hdydrocarbon reservoirs. The idea is that "information spread" along the stream lines of the hydrocarbon reservoir is fast and information spread by diffusion is slow. The result is that grid blocks that are connected by stream lines are "closer" to each other than grid blocks that are not connected in that way. Then again a function with compact support is applied and the result is used within the Kalman update.

Results obtained with these localization approaches show that they are very useful and can mitigate the above mentioned long-range spurious correlations significantly.

8.2.2 SEIK Filter

The singular evolutive interpolated Kalman (SEIK) filter is similar to the SEEK filter in section 8.1.3.3, only that the construction of the tangential space is now performed by a sampling procedure (which is why we consider it a stochastic filter). This sampling procedure is expected to reduce the approximation error of the attractor by a tangential space [Pha97].

This filter bears some resemblance of the EnKF and in [Pha01] some modifications to the original EnKF are proposed (use of a forgetting factor, adaptive scaling of the observation error covariance matrix) which were postulated by the author for his own filter developments in earlier articles. Especially the introduction of a forgetting factor has a huge positive impact on the filter performance, at least in the case of the strongly non-linear Lorentz model used in the article.

8.2.3 Sequential Monte Carlo Filters

In this literature review all stochastic filters that do *not* employ a Gaussian assumption on the involved error statistics are called *sequential Monte Carlo* (SMC) or *particle filters*. Note that EnKF (section 8.2.1) and SEIK (section 8.2.2) can very well be considered as special cases of these methods as they only involve additional asspumptions, but the consequences that come from these assumptions make an impact on the filter algorithms which is huge enough to justify the separate treatment. [Pha01, p. 1197] even thinks that "the particle filter has nothing to do with the

[1] What "somehow" actually means is discussed below.

Kalman filter". While being able to understand the point of this statement I consider it a little bit too strong: the theoretical relation is obviously there; the impact of the Gaussian assumption on the resulting algorithm is quite large, though.

SMC filters do not make any assumptions on the involved error statistics. They are, at least in the limit of infinitely many ensemble members, able to capture all non-linearities of the model correctly: every ensemble member is computed forward in time using the full non-linear model, like in the EnKF. But the updating procedure is different: SMC filters now only *weight* the ensemble members according to the data likelihood instead of directly modifying the members by a linear combination that reflects the data likelihood. This weighting leads to filter divergence very quickly, where one ensemble member attracts almost full weight while the others approach an almost zero weighting. The solution is an additional re-sampling stage, which is not present in EnKF.

Probably the most well-known SMC filter is the *bootstrap filter* published by N. J. Gordon, D. J. Salmond and A. F. M. Smith in 1993 [GSS93].

Re-Sampling Problems in SMC Re-sampling the particles after weighting from a discrete distribution can be a problem if the system noise is low: new particles will be very close to each other if drawn from the same parent particle. [Pha01, p. 1196] proposes to take a certain continuous replacement distribution which approximates the discrete one.

8.2.3.1 Unscented Particle Filter

The *unscented particle filter* [vdMDdFW00] is a particle filter that uses the unscented Kalman filter (section 8.1.3.4) to create the importance proposal distribution. According to the author it should outperform standard particle filters, extended Kalman filters and unscented Kalman filters.

8.3 Further Reading

Algorithms that not only incorporate information into the current analysis of the state but also backwards in time are called "smoothers": they smooth the whole time series and not only update the current state. Due to their similarity to filters they were not treated separately in this review, but a recent overview of smoothing algorithms for state space models is given by [BDM04].

[NJC06] performs a comparison of EnKF, EnKS and representer method. The sequential methods seem to be more suitable for forecasts, while the variational method is better for analysis.

Chapter 9

Discussion

This discussion covers the last three chapters and points out important aspects, similarities and differences for the three different views and their methods.

9.1 Connections of the Views

Especially when the Gaussian assumption comes into play, all three views move together. They all have some methods in stock which deal with least squares-like problems. For example, according to [Neu98, p. 17], Wiener filtering can be seen as "probably the earliest application of regularization techniques". The Wiener filter originally was created by Norbert Wiener for optimal noise reduction of a signal. Wiener introduced additional knowledge by auto- and cross-correlation functions of the noise and the signal and from this computed an optimal (in a least squares sense) filter function to remove the noise from the signal.

So even at this early time we see connections between regularisation and filtering. And recursive Bayesian estimation, as filtering is also sometimes called, makes the connection from the state space view to the probabilistic view – there even for non-Gaussian problems.

9.2 Practicability

The probabilistic view, together with its methods, is certainly the most rigorous one. It gives the reader much insight into the problem of inverse modeling, and especially [Tar04] is worth reading. But apart from least squares, essentially only sampling methods are used. This makes the view practially impossible for larger problems.

The other two views are more pragmatic: they try to find at least an approximate solution, which can be practically used, even though it may not be the best possible one. But [Tar04] even postulates that sampling methods should be used as a "movie strategy": he says that we have to look "at a movie of realizations" (meaning pictures of realisations) of the prior and posterior distributions, and verify that they are adequate. Only then we can use these realisations to compute statistics on them. For him, looking at the "best realisation" or the "mean realisation", even with accompanied error analysis, is not a proper way of dealing with inverse problems. But this is what is done, especially in the ill-posed optimisation problem view. While I must admit that [Tar04] is right in a rigorous sense, sometimes one has to be more pragmatic than him and use what one's got to make any progress.

9.3 Parameter Estimation using Sequential Methods

The idea to treat uncertain parameters in the same way as state variables is quite popular, but one has to keep in mind that almost all simulators used to compute the model forecasts were

not designed to handle timevariant parameters – which essentially is what results when modifying parameters during a simulation. The partial differential equations forming the basis of such a simulator simply do not handle this. Despite the success of this method, the implications of this are quite unclear and the issue is simply not mentioned and neglected.

Chapter 10

Summary and Outlook

The trend towards stochastic methods, let it be probabilistic methods, sequential methods or optimisation methods, cannot be overlooked and is mainly due to the ever increasing computing power. Today even realistically sized simulation models can be used in stochastic filters or sampling methods. Until lately, mainly stochastic optimisation strategies (giving only one, somehow optimal result) have been used. This trend towards even more stochastic methods shows that the solution of the fully probabilistic formulation will come in reach, at least for smaller problems.

Another important trend that can be seen when looking at the different methods is that for complex problems, stochastic population (or ensemble) based algorithms are quite popular in the state space view and the ill-posed optimisation problem view at the moment. Having different features and preferences, it is interesting and also challenging to combine aspects of methods coming from both views into new, more specialised (or even more general) hybrid methods. For the important aspect of uncertainty quantification, it is very interesting to think about the probabilistic view. The sampling methods used there may not be very usable, but due to the connection to the state space view and its quite efficient methods, aspects of a rigorous uncertainty quantification may also be possible to be embedded into such hybrid methods.

Bibliography

[AM79] Brian D. O. Anderson and John B. Moore. *Optimal filtering*. Prentice–Hall information and system sciences series. Prentice–Hall, Englewood Cliffs, NJ., 1979.

[AN06] Elkin Rafael Arroyo Negrete. Continuous reservoir model updating using an ensemble Kalman filter with a streamline-based covariance localization. Master's thesis, Texas A&M University, 2006.

[AO08] Chinedu Agbalaka and Dean Oliver. Application of the EnKF and Localization to Automatic History Matching of Facies Distribution and Production Data. *Mathematical Geosciences*, 40(4):353–374, May 2008.

[AS72] D. Alspach and H. Sorenson. Nonlinear Bayesian estimation using Gaussian sum approximations. *Automatic Control, IEEE Transactions on*, 17(4):439–448, Aug 1972.

[Bac00] Alexander Backlund. The definition of system. *Kybernetes*, 29(4):444–451, 2000.

[Bal00] Helmut Balzert. *Lehrbuch der Software-Technik - Software-Entwicklung*. Spektrum Akad. Vlg., 2. auflage edition, 2000.

[BD05] John Baird and Clint Dawson. The representer method for data assimilation in single-phase darcy flow in porous media. *Computational Geosciences*, 9(4):247–271, December 2005.

[BDM04] Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state-space models. Technical Report CUED/F-INFENG/TR.498, Cambridge University Engineering Department, 2004.

[BK08] Frank Bauer and Stefan Kindermann. The quasi-optimality criterion for classical inverse problems. *Inverse Problems*, 24(3):035002–, 2008.

[Bro70a] C. G. Broyden. The convergence of a class of double-rank minimization algorithms: 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, March 1970.

[Bro70b] C. G. Broyden. The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. *IMA Journal of Applied Mathematics*, 6(3):222–231, September 1970.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BVA07] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6(4):467–484, 2007.

[BVA08] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1):109–124, March 2008.

[Cau47] A. Cauchy. Méthode générale pour la resolution des systems d'équations simultanées. *Comp. Rend. Acad. Sci.*, 25:536–538, 1847.

[CG95] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[Dak65] R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.

[Dan02] George B. Dantzig. Linear programming. *Operations Research*, 50:42–47, 2002.

[DBM03] Julien Diard, Pierre Bessière, , and Emmanuel Mazer. A survey of probabilistic models, using the bayesian programming methodology as a unifying framework. In *The Second International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, 2003.

[DdFG01] Arnaud Doucet, Nando de Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science. Springer, 2001.

[DES98] Peter Deuflhard, Heinz W. Engl, and Otmar Scherzer. A convergence analysis of iterative methods for the solution of nonlinear ill-posed problems under affinely invariant conditions. *Inverse Problems*, 14(5):1081–1106, 1998.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[DS04] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[EHN00] Heinz W. Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*. Kluwer, 2000.

[Esp05] Jairo Espinosa. *Fuzzy logic, identification and predictive control*. Advances in industrial control. Springer, London, 2005.

[Eve07] Geir Evensen. *Data Assimilation - The Ensemble Kalman Filter*. Springer, 2007.

[FGS+94] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical Physics Letters*, 219(5-6):343–348, March 1994.

[FHH07] Elana J. Fertig, John Harlim, and Brian R. Hunt. A comparative study of 4d-var and a 4d ensemble kalman filter: perfect model simulations with lorenz-96. *Tellus A*, 59(1):96–100, 2007.

[FL97] Urban Forssell and Peter Lindskog. Combining semi-physical and neural network modeling: An example of its usefulness. Technical report, Department of Electrical Engineering, Linköping University, 1997.

[Fle70] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, March 1970.

[FP63] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimisation. *The Computer Journal*, 6:163–168, 1963.

[FR64] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, February 1964.

[GCC08] A. Gaspar-Cunha and J. Covas. Robustness in multi-objective optimization using evolutionary algorithms. *Computational Optimization and Applications*, 39(1):75–96, January 2008.

[Gev06] Michel Gevers. A personal view of the development of system identification: A 30-year journey through an exciting field. *Control Systems Magazine, IEEE*, 26(6):93–105, Dec. 2006.

[GG84] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–742, 1984.

[GKL01] Zong Woo Geem, Joong Hoon Kim, and G.V. Loganathan. A new heuristic optimization algorithm: Harmony search. *SIMULATION*, 76(2):60–68, February 2001.

[Gol70] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[Gom58] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

[Gom02] Ralph E. Gomory. Early integer programming. *Operations Research*, 50:78–81, 2002.

[GSS93] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F Radar and Signal Processing*, volume 140, pages 107–113, April 1993.

[Ham06] K. Hamacher. Adaptation in stochastic tunneling global optimization of complex potential energy landscapes. *Europhysics Letters*, 74:944–950, 2006.

[Has70] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[HM98] P. L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble kalman filter technique. *Monthly Weather Review*, 126:796–811, March 1998.

[HM01] P. L. Houtekamer and Herschel L. Mitchell. A sequential ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, January 2001.

[Hou70] Alston S. Householder. *The Numerical Treatment of a Single Nonlinear Equation*. McGraw-Hill, 1970.

[HS52] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, December 1952.

[HW03] Simon S Haykin and Bernard Widrow. *Least-Mean-Square Adaptive Filters*. Wiley-Interscience, 2003.

[HWS01] Thomas M. Hamill, Jeffrey S. Whitaker, and Chris Snyder. Distance-dependent filtering of background error covariance estimates in an ensemble kalman filter. *Monthly Weather Review*, 129(11):2776–2790, November 2001.

[Ise92] Rolf Isermann. *Identifikation dynamischer Systeme 1 - Grundlegende Methoden*. Springer, 2nd edition edition, 1992.

[Ise05] Rolf Isermann. *Mechatronic Systems - Fundamentals*. Springer, 2005.

[Jat06] Ravindra V. Jategaonkar. *Flight vehicle system identification: Atime domain methodology*, volume 216 of *Progress in astronautics and aeronautics*. American Institute of Aeronautics and Astronautics, Reston, Va., 2006.

[JGJS99] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

[JU97] Simon Julier and Jeff Uhlmann. A new extension of the kalman filter to nonlinear systems. In *SPIE AeroSense Symposium, April 21-24*, 1997.

[JU04] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, Mar 2004.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, 82(Series D):35–45, March 1960.

[Kal97] Barbara Kaltenbacher. Some newton-type methods for the regularization of nonlinear ill-posed problems. *Inverse Problems*, 13(3):729–753, 1997.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, December 1984.

[KGV83] S. Kirkpatrick, Jr. Gelatt, C. D., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[Kha79] Leonid Genrikhovich Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[Kir96] Andreas Kirsch. *An introduction to the Mathematical Theory of Inverse Problems*, volume 120 of *Applied Mathematical Sciences*. Springer, 1996.

[Kit87] Genshiro Kitagawa. Non-gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82(400):1032–1041, 1987.

[Lan95] Börje Langefors. *Essays on Infology: Summing Up and Planning for the Future*. Studentlitteratur, Lund, 1995.

[LD60] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[Lju06] Lennart Ljung. *System identification: Theory for the user*. Prentice Hall information and system sciences series. Prentice Hall PTR, Upper Saddle River, NJ, 2. ed., nachdr. edition, 2006.

[LLW00] Jun S. Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[LM85] A. V. Levy and A. Montalvo. The tunneling algorithm for the global minimization of functions. *SIAM J. Sci. and Stat. Comput.*, 6(1):15–29, January 1985.

[LN89] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[LPR07] Shuai Lu, Sergei V. Pereverzev, and Ronny Ramlau. An analysis of tikhonov regularization for nonlinear ill-posed problems under a general smoothness assumption. *Inverse Problems*, 23(1):217–230, 2007.

[LZ07] Hui Li and Qingfu Zhang. Comparison between nsga-ii and moea/d on a set of multiobjective optimization problems with complicated pareto sets. Technical Report CES-476, Department of Computing and Electronic Systems, University of Essex, 2007.

[Mar75] J. H. Marchal. On the concept of a system. *Philosophy of Science*, 42(4):448–468, 1975.

[MKT07] L. Melara, A. Kearsley, and R. Tapia. Augmented lagrangian homotopy method for the regularization of total variation denoising problems. *Journal of Optimization Theory and Applications*, 134(1):15–25, July 2007.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[MT02] Klaus Mosegaard and Albert Tarantola. Probabilistic approach to inverse problems, November 2002.

[MU49] Nicholas Metropolis and Stanley Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949.

[Nel01] Oliver Nelles. *Nonlinear system identification: From classical approaches to neural networks and fuzzy models*. Engineering online library. Springer, Berlin, 2001.

[Neu98] Arnold Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40:636–666, 1998.

[NJC06] Hans E. Ngodock, Gregg A. Jacobs, and Mingshi Chen. The representer method, the ensemble kalman filter and the ensemble kalman smoother: A comparison study using a nonlinear reduced gravity ocean model. *Ocean Modelling*, 12:378–400, 2006.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

[Noc80] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, July 1980.

[Pas02] K.M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems Magazine, IEEE*, 22(3):52–67, 2002.

[PGK+06] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm – a novel tool for complex optimisation problems. In *Proceedigns of the IPROMS 2006*, 2006.

[Pha97] Dinh Tuan Pham. Dimension, predictability and reduced rank kalman filtering in data assimilation. In *Proceedings of the Third Bilateral French-Russian Conference: Predictability of Atmospheric and Oceanic Circulations*. Institute Franco-Russe A. M. Lyapunov (INRA - Moscow State University), 1997.

[Pha01] Dinh Tuan Pham. Stochastic methods for sequential data assimilation in strongly nonlinear systems. *Monthly Weather Review*, 129(5):1194–1207, May 2001.

[PJHJH07] Justyna Przybysz-Jarnut, Remus Hanea, Jan-Dirk Jansen, and Arnold Heemink. Application of the representer method for parameter estimation in numerical reservoir models. *Computational Geosciences*, 11(1):73–85, March 2007.

[Pow77] M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241–254, December 1977.

[PVCR98] Dinh Tuan Pham, Jacques Verron, and Marie Christine Roubaud. A singular evolutive extended kalman filter for data assimilation in oceanography. *Journal of Marine Systems*, 16(3-4):323–340, October 1998.

[Rei79] D. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, Dec 1979.

[RJK+00] F. Rabier, H. Järvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons. The ecmwf operational implementation of four-dimensional variational assimilation. i: Experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1143–1170, 2000.

[Sch75] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universität Berlin, 1975.

[Sha70] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970.

[Sil01] William Silvert. Modelling as a discipline. *International Journal of General Systems*, 30(3):261–282, 2001.

[Sky07] Lars Skyttner. *General Systems Theory*. World Scientific Publishing Company, 2. ed., reprinted edition, 2007.

[Spa62] H. A. Spang. A review of minimization techniques for nonlinear functions. *SIAM Review*, 4(4):343–365, 1962.

[Tar04] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2004.

[vB68] Ludwig von Bertalanffy. *General System Theory: Foundations, Development, Applications*. George Braziller, Inc., 1968.

[vdMDdFW00] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The unscented particle filter. Technical report, Cambridge University Engineering Department, 2000.

[VK99] F. Vandenberghe and Y.-H. Kuo. Introduction to the mm5 3d-var data assimilation system: Theoretical basis. Technical report, National Center for Athmospheric Research, Boulder, 1999.

[Wei] Eric W. Weisstein. Brachistochrone Problem. From MathWorld–A Wolfram Web Resource.

[WH99] W. Wenzel and K. Hamacher. Stochastic tunneling approach for global minimization of complex potential energy landscapes. *Physical Review Letters*, 82(15):3003–3007, April 1999.

[Wik08] Wikipedia. Cybernetics, 2008.

[Yuk07] L. Yukhno. On modification of certain methods of the conjugate direction type for solving systems of linear algebraic equations. *Computational Mathematics and Mathematical Physics*, 47(11):1737–1744, November 2007.

[ZLT01] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

Technische Universität Braunschweig
Informatik-Berichte ab Nr. 2006-02

2006-02	T. Mücke, B. Florentz, C. Diefer	Generating Interpreters from Elementary Syntax and Semantics Descriptions
2006-03	B. Gajanovic, B. Rumpe	Isabelle/HOL-Umsetzung strombasierter Definitionen zur Verifikation von verteilten, asynchron kommunizierenden Systemen
2006-04	H. Grönniger, H. Krahn, B. Rumpe, M. Schindler, S. Völkel	Handbuch zu MontiCore 1.0 - Ein Framework zur Erstellung und Verarbeitung domänenspezifischer Sprachen
2007-01	M. Conrad, H. Giese, B. Rumpe, B. Schätz (Hrsg.)	Tagungsband Dagstuhl-Workshops MBEES: Modellbasierte Entwicklung eingebetteter Systeme III
2007-02	J. Rang	Design of DIRK schemes for solving the Navier-Stokes-equations
2007-03	B. Bügling, M. Krosche	Coupling the CTL and MATLAB
2007-04	C. Knieke, M. Huhn	Executable Requirements Specification: An Extension for UML 2 Activity Diagrams
2008-01	T. Klein, B. Rumpe (Hrsg.)	Workshop Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen, Tagungsband
2008-02	H. Giese, M. Huhn, U. Nickel, B. Schätz (Hrsg.)	Tagungsband des Dagstuhl-Workshopss MBEES: Modellbasierte Entwicklung eingebetteter Systeme IV
2008-03	R. van Glabbeek, U. Goltz, J.-W. Schicke	Symmetric and Asymmetric Asynchronous Interaction
2008-04	R. van Glabbeek, U. Goltz, J.-W. Schicke	On Synchronous and Asynchronous Interaction in Distributed Systems
2008-05	M. V. Cengarle, H. Grönniger B. Rumpe	System Model Semantics of Class Diagrams
2008-06	M. Broy, M. V. Cengarle, H. Grönniger B. Rumpe	Modular Description of a Comprehensive Semantics Model for the UML (Version 2.0)
2008-07	C. Basarke, C. Berger, K. Berger, K. Cornelsen, M. Doering J. Effertz, T. Form, T. Gülke, F. Graefe, P. Hecker, K. Homeier F. Klose, C. Lipski, M. Magnor, J. Morgenroth, T. Nothdurft, S. Ohl, F. Rauskolb, B. Rumpe, W. Schumacher, J. Wille, L. Wolf	2007 DARPA Urban Challenge Team CarOLO - Technical Paper
2008-08	B. Rosic	A Review of the Computational Stochastic Elastoplasticity
2008-09	B. N. Khoromskij, A. Litvinenko, H. G. Matthies	Application of Hierarchical Matrices for Computing the Karhunen-Loeve Expansion
2008-10	M. V. Cengarle, H. Grönniger B. Rumpe	System Model Semantics of Statecharts
2009-01	H. Giese, M. Huhn, U. Nickel, B. Schätz (Herausgeber)	Tagungsband des Dagstuhl-Workshops MBEES: Modellbasierte Entwicklung eingebetteter Systeme V
2009-02	D. Jürgens	Survey on Software Engineering for Scientific Applications: Reuseable Software, Grid Computing and Application
2009-03	O. Pajonk	Overview of System Identification with Focus on Inverse Modeling